Федеральное государственное образовательное бюджетное учреждение высшего профессионального образования «Сибирский государственный университет телекоммуникаций и информатики»
(ФГОБУ ВПО «СибГУТИ»)

А.Е. Костюкович

Организация сети для услуг управления

Практикум

Новосибирск 2012

Костюкович А.Е..

Аннотация.

Практикум к Контрольной работе для дисциплины "Эксплуатация и техническое обслуживания СПС". Может быть также использован в процессе изучения дисциплин "Мультисервисные сети" и "Сети связи". В практикуме приводятся основные сведения, касающиеся проектирования ресурсов для обеспечения обмена управляющей информацией между менеджером и агентами по протоколу SNMP. Рассматриваются также вопросы кодирования управляющей информации, способы описания свойств управляемых объектов, основы стека протоколов TCP-UDP/IP, служащего транспортной основой для протокола SNMP. Для лучшего усвоения материала, студенту предлагается произвести расшифровку управляющих сообщений протокола SNMP, а также заголовков транспортных протоколов.
Во второй части работы рассматривается проект ресурсов транспортной сети для пропуска управляющего трафика между менеджером и агентами.

Кафедра АЭС
Ил. 15, список лит. - 13

Рецензент – Мелентьев О.Г.

По направлению – 210400 - Телекоммуникации

Утверждено редакционно-издательским советом СибГУТИ
в качестве методических указаний

Оглавление	
	Стр.

1. Принципы обмена управляющей информацией

Управляющая информация (тип **M – management**) является одним из важнейших типов информации наряду с пользовательским типом информации (тип **U – user**) и информацией сигнализации (тип **C – control**).

Среди видов информации типа M выделим следующие виды:

- информация о состоянии управляемых объектов (status – up/down, alarm,…)
- информация о нагрузке (обслуженной, отброшенной, …)
- информация биллинга (CDR-файлы)
- информация конфигурации управляемых объектов (команды менеджера и рапорты-отчеты агентов)

Указанные виды управляющей информации обладают различными свойствами:

- характером нагрузки, создаваемой на транспортную сеть
- моделью, описывающей статистические свойства этой нагрузки
    - детерминированные модели
    - стохастические модели
- видом распределения и параметрами модели:
    - интенсивностью поступления требований
    - неравномерностью (пачечностью) поступающего трафика

Источниками управляющей информации являются:

- Управляемые объекты, передающие информацию либо по запросу менеджера, либо по изменению состояния объекта
- Управляющие системы (менеджеры), генерирующие управляющую информацию (запросы, команды,…) либо автоматически (по заранее составленному оператором расписанию), либо в ручном режиме (команды MML, вводимые операторами OMC)

Характер и направление обмена управляющей информацией определяется моделью отношений «Менеджер – Агент». Менеджер – это ПО, представляющее управляющую систему, рассылающую запросы/команды Агентам, в качестве которых выступает ПО, распределенное по управляемым объектам.

Подробнее о модели и принципах обмена Менеджер – Агент смотрите в разделе 2.1 данных методических указаний, а также в пособии [1].

Здесь отметим наиболее важные моменты в контексте данного проекта:

- агенты размещаются непосредственно на территориально распределенных управляемых объектах
- информация от агентов к менеджерам доставляется по той же мультисервисной транспортной сети, по которой перевозится и информация типов U и C
- для изоляции управляющего трафика в целях безопасности – в транспортной сети организуются либо выделенные каналы уровня L1, либо – виртуальные частные сети (VPN)

1.1. Архитектура современных инфокоммуникаций

В рамках создания единой Глобальной Информационной Инфраструктуры (GII, Y.100…Y.120) была предложена уровневая архитектура современных инфокоммуникаций, вошедшая впоследствии в архитектурную концепцию построения Сетей Следующего Поколения (NGN).

Данная архитектура представлена на рис. 1.1

Рис. 1.1 – Уровневая архитектура NGN

Уровневая архитектура сети нового поколения (NGN) включает:

- **Мультисервисную транспортную платформу**. Поддерживает нижние 3 уровня OSI. В свою очередь транспортная платформа включает следующие уровни:
  - **уровень ядра транспортной сети (Core Network – CN),** реализуемый на базе технологий мультисервисных транспортных сетей (на данный момент наиболее проработаны технологии ATM, IP/MPLS/all, IP/VLAN/Ethernet)
  - **уровень сетей доступа (Access Network – AN),** отличающийся многообразием используемых технологий. Наиболее распространенными в настоящее время являются следующие технологии доступа – xDSL, ETTH, Wi-Fi, Wi-Max, PON.
  - **пограничный уровень** (понятие границы – достаточно условное и определяется функционально и организационно, т.е. операторами, владеющими ядром сети и сетью доступа)
- **Платформу управления соединениями и сигнализации**, реализуемую на базе новых программно-аппаратных комплексов, за которыми закреплено название Softswitch (гибкая система управления коммутацией)
- **Платформу серверов**, обеспечивающих необходимый набор информационных услуг, включая услуги по размещению, хранению и обработке разнообразного информационного контента, **а также вспомогательные услуги, к которым можно отнести услуги управления сетью и сервисами**.

Для взаимодействия между этими платформами используются универсальные открытые интерфейсы.

Вследствие того, что информационные услуги предоставляются на удалении от серверов, т.е. для доступа к ним требуется транспортировка информации (коммуникационные услуги), перечисленные выше услуги (транспортные и информационные) часто называют одним термином:
ИНФОКОММУНИКАЦИОННЫМИ услугами!

Инфокоммуникационные услуги предоставляются на базе узлов (серверов) служб (Services Node – SN), принадлежащих поставщикам услуг (сервис-провайдерам – SP). Данные узлы в процессе предоставления услуги могут выполнять одну или несколько функций, например, функцию аутентификации, функцию выполнения логики услуги, функцию хранения, обработки и поиска информации и т.д.

В процессе предоставления инфокоммуникационных услуг может потребоваться взаимодействие узлов служб, реализующих разные функции. Как следствие, узлы служб одного поставщика услуг могут образовывать **платформы услуг**.

Платформы услуг разных поставщиков услуг, предоставляющих однотипные услуги, могут объединяться в **сети услуг** (**Services Networks**).

Сети услуг – это виртуальные логические сети, представляющие собой некоторые надстройки над транспортными сетями, например:

- Интеллектуальная сеть,
- **Сеть услуг управления, например, на базе технологий TMN или SNMP**,
- Сеть WWW
- Сеть IPTV и т.д.

Узлы служб должны иметь стандартные интерфейсы, а услуги должны предоставляться по стандартизованным сценариям.

1.2. Характеристика услуг управления.

Для правильного проектирования ресурсов транспортной сети необходимо знать характеристики услуг управления, которые рассматриваются в данном проекте как часть информационных услуг или вспомогательных услуг (Middle Ware).

Представим рисунок 1.1 для услуг управления (мониторинг, статистика, биллинг, обеспечение безопасности,…) – см. рис. 1.2:

Рисунок 1.2 – Архитектура Сети услуг управления

Услуги управления сетью имеют следующие особенности:

1. Объектами управления (клиентская часть) является как **терминалы и серверы**, так и **оборудование и ПО собственно транспортной сети и сети доступа**.
2. Объектами управления являются также **параметры NP и QoS** (статистика).

Транспортная сеть должна проектироваться как в расчете на пропуск **пользовательского контента** (**в первую очередь**), так и на пропуск **служебного контента**, к которому, в частности относится и управляющий контент.

Рассмотрим основные свойства информационных услуг, в первую очередь – услуг управления.

Все услуги по управлению сетью и сервисами согласно рек. ITU-T X.700 и M.3010 классифицируются на следующие **5 функциональных областей**:

1. **FM (fault management)** – Управление устранением последствий отказов
2. **CM (configuration management)** – Управление конфигурацией сети и услуг
3. **AM (accounting management)** – Управление расчетами (биллинг)
4. **PM (performance management)** – Управление рабочими характеристиками сети
5. **SM (security management)** – Управление безопасностью

Перечисленные функциональные области иногда совместно обозначаются как **FCAPS** (по первым буквам англоязычных обозначений).

Требования, предъявляемые со стороны пользователей и разработчиков к реализации перечисленных функциональных областей, отражаются в соответствующих спецификациях через функции управления системами (Systems Management Function, SMF), которые реализуются за счет сервисов или услуг управления на соответствующем уровне модели ВОС.

Общее требование ко всем функциональным областям управления состоит в том, что **вид управления** определяется **причиной происходящих в сети событий**, т.е. управление инициируется посредством сообщений о событиях в сети.

Другими словами, управление не происходит самопроизвольно, переход от одного процесса или процедуры управления к другой обусловлен воздействием какого-то внутреннего или внешнего события. Под внешним событием понимается, например, потеря электропитания, пожар или затопление помещения. Под внутренним событием понимается, к примеру, истечение предельного времени выполнения определенного задания или теста, сбой программного обеспечения, выход из строя функционального модуля.

Рассмотрим подробнее, что включает в себя каждая функциональная область управления.

Управление неисправностями (fault management – FM) характеризуется, прежде всего, функцией генерации специфических сообщений о неисправностях — так называемых тревог (alarms). При этом осуществляется регистрация источника сообщений об ошибке и начинается тестирование сетевых ресурсов с тем, чтобы идентифицировать и контролировать неисправности. При управлении неисправностями необходимо предпринимать действия по наблюдению за неисправностями (анализ, фильтрация и корреляция сообщений о неисправностях), выполнять тестирование неисправного ресурса, обеспечивать локализацию неисправности, а также исправлять неисправности.

Управление устранением последствий отказов включает следующие действия:

- обнаружение отказа
- регистрация отказа
- трассировка и идентификация отказа
- изоляцию (локализация) отказа
- исправление отказа
- диагностика и тестирование после исправления отказа.

Основное требование к управлению неисправностями — это наличие операций (процедур, действий) управления, инициируемых определенными сетевыми событиями.

Управление расчетами (accounting management – AM) за услуги связи — это совокупность процедур учета информации о количестве и объемах оказанных услуг связи и обработки зафиксированных данных в целях подготовки счетов с начислениями за услуги связи.

Управление расчетами включает:

1. Измерение использования сетевых услуг и ресурсов:
2. Тарификация и ценообразование:
3. Сбор и финансирование:
4. Управление предприятием:

Ключевые требования к управлению расчетами — наличие операций, зависящих от событий, в особенности регистрация услуг и основные правила регистрации использования услуг связи или сетевого ресурса.

Управление конфигурацией (configuration management – CM) обеспечивает инициализацию (запуск), установку и обеспечение функционирования оборудования связи. Это позволяет осуществлять в едином комплексе работы по пуско-наладке оборудования и передачу информацию о состоянии оборудования по запросу администратора сети. Появляется техническая возможность обеспечивать средства технического учета оборудования и поддерживать уведомления об изменениях конфигурации оборудования через соответствующие сообщения.

Управление конфигурацией включает в себя:

1. Планирование и разработка сети:
2. Монтаж и установка оборудования:
3. Планирование услуг и контрактов с абонентами:
4. Обеспечение услуг и сетевых ресурсов:
5. Состояние и контроль:

Основные требования к управлению конфигурацией: наличие операций над объектами управления; контроль изменений конфигурации; контроль первичного состояния ресурсов сети; представление связей и взаимоотношений между объектами управления в форме, понятной для разработчика системы управления и пользователя; возможность планирования сети; управление временем; распределение программного обеспечения и наличие средств восстановления системы.

Управление рабочими характеристиками и производительностью (performance management – PM) сети предполагает наличие и доступность информации управления с целью определения технического состояния сети и загрузки системы связи при естественных и искусственных, т.е. смоделированных условиях. Управление рабочими характеристиками сети поддерживает совокупную информацию об эффективности работы сети, которая поступает периодически, обеспечивая тем самым статистику работы сети и позволяя планировать различные управляющие воздействия.

Для управления характеристиками сети необходим доступ к большому количеству сетевой информации. При этом особенно важна проблема обеспечения степени воздействия на управляемую сеть. Как правило, желательно, чтобы каждое отдельно взятое воздействие было минимальным. Ключевое требование для данного вида управления — способность преобразования первичной информации о сетевой ситуации в формальные показатели (с учетом пороговых значений этих показателей) в соответствующие периоды времени. К такого рода задачам относится, например, задача преобразования сведений о количестве и продолжительности поступивших вызовов в данные о нагрузке канала связи и последующего вывода о наличии перегрузки.

Даже из такого простейшего примера следует, что при управлении производительностью необходима процедура периодического агрегирования (обобщения) информации об эффективности работы сети для выявления тенденций развития сетевой ситуации и планирования пропускной способности. Соответственно необходимы средства планирования для регулярного сбора информации о работе сети, а также возможность определения времени получения отклика о состоянии объектов управления на сети.

Управления рабочими характеристиками/производительностью включает в себя:

1. Обеспечение QoS:
2. Мониторинг рабочих характеристик:
3. Контроль параметров управления:
4. Анализ рабочих характеристик:

Управление безопасностью (security management – SM) затрагивает два аспекта защиты систем:

- управление собственно безопасностью (management of security) — способность контроля и управления средствами защиты и своевременного сообщения об угрозах безопасности или нарушениях безопасности сетей и средств связи;
- безопасность управления (security of management) — возможность опознавания пользователей системы управления и соответствующих прикладных программ, что гарантирует конфиденциальность и целостность обмена информацией управления и предотвращает несанкционированный доступ к информации управления. Услуги установления подлинности, обеспечения целостности данных и конфиденциальности являются общими для всех прикладных программ ВОС и затрагивают все процедуры управления.

Управление безопасностью включает в себя следующие функции:

1. Предупреждение:
2. Обнаружение:
3. Защита от распространения и восстановление:
4. Администрирование:

Ключевые требования при управлении безопасностью согласно модели BOC — это поддержка аварийных сообщений о безопасности, средства для проведения аудита безопасности и средства управления доступом.

При централизованном выполнении действий со стороны менеджера, расположенного в Центре Технической Эксплуатации (ЦТЭ – ОМС), **обмен управляющей информацией между менеджером и агентами осуществляется по СПД**.

Этот обмен требует выделения определенных сетевых ресурсов (пропускной способности, объема буферной памяти в сетевых узлах СПД, производительности сетевых устройств – маршрутизаторов, коммутаторов и др.), а также серверных ресурсов.

В данном проекте рассчитываются ресурсы для обеспечения услуг управления сетью.

В состав проектируемых услуг управления входят:

- **коммуникационные услуги** (**транспортные**), заключающиеся в «перевозке» управляющей информации по транспортной сети,
- **информационные услуги**, заключающиеся в обеспечении обработки, хранения, поиска управляющей информации в распределенной системе «Менеджер-Агент».

Таким образом, необходимо выполнить **две части проекта**:

- **Разработка надстройки, представляющей управляющую платформу для поддержки услуг по управлению сетью**
  - Кратко описать принципы взаимодействия «Менеджер-Агент» по протоколу SNMP
  - Составить схему проектируемой сети, с указанием менеджера и агентов, распределенных по объектам транспортной сети
  - Привести стеки протоколов для обмена управляющей информацией между менеджером и агентом по сети IP
  - Привести расшифровку сообщений протокола SNMP согласно варианту задания. В процессе выполнения этого пункта познакомиться:
    - С типами управляющей информации, содержащейся в MIB
    - Со структурой этой информации и правилами ее кодирования
    - Оценить объем этой информации в одном SNMP-сообщении
  - Оценить объем нагрузки на транспортную сеть от агентов и менеджеров
  - Оценить пропускную способность сетевых интерфейсов, в точках подключения менеджера и агентов к транспортной сети
  - Выбрать ПО менеджера (систему управления/мониторинга)
- **Проект ресурсов транспортной сети, обеспечивающей «перевозку» управляющей информации**
  - Проект VPN для услуг управления
  - Определение класса обслуживаемого трафика
  - Организация требуемой пропускной способности.

2. Часть 1. Проект управляющей платформы

В этой части проекта Вы должны разработать структурную схему сети, обеспечивающей обмен управляющей информацией между менеджером и агентами, включая сервисную платформу (менеджера и агентов) и транспортную мультисервисную сеть в части ресурсов, обеспечивающих перенос управляющей информации с необходимым уровнем качества (безопасность, класс обслуживания, пропускная способность).

Теория по SNMP-технологии

2.1. Принципы взаимодействия «Менеджер-Агент» по протоколу SNMP

Для централизованного управления сетевыми элементами, используется схема Менеджер-Агент, причем программно-аппаратные средства менеджера инсталлируются в центре управления сетью, а программно-аппаратные средства агентов распределены территориально по управляемым объектам. Для передачи управляющей информации используются различные транспортные сети.

Агенты делают информацию доступной для систем управления сетями (Network Management Systems – NMS) **с помощью управляющих протоколов**.

В качестве управляющего протокола, обеспечивающего основные функции управления, **в данном пособии рассматривается протокол верхнего уровня – SNMP** (Simple Network Mangement Protocol – простой протокол управления сетью).

Программа сервера, называемая сетевым менеджером, осуществляет виртуальные соединения с программой, которая называется SNMP-агентом. SNMP-агент расположен на удаленном сетевом устройстве и предоставляет информацию менеджеру о состоянии данного устройства.

В данном пособии рассматривается пример обмена по сети, построенной на базе стека протоколов TCP-UDP/IP. В качестве сети доставки рассмотрен вариант локальной сети на базе протоколов сети Ethernet.

Эта модель представлена на рис. 1. Управляемым объектом в SNMP может быть некоторый атрибут, т.е. характеристика объекта, имеющая какую-либо ценность с точки зрения управления, например, количество переданных пакетов через определенный интерфейс, состояние управляемого объекта, физический адрес сетевого интерфейса, элементы записи в маршрутной таблице и т.д.

Рассмотрим детально схему обмена управляющей информацией Менеджер-Агент (рисунок 2.1).

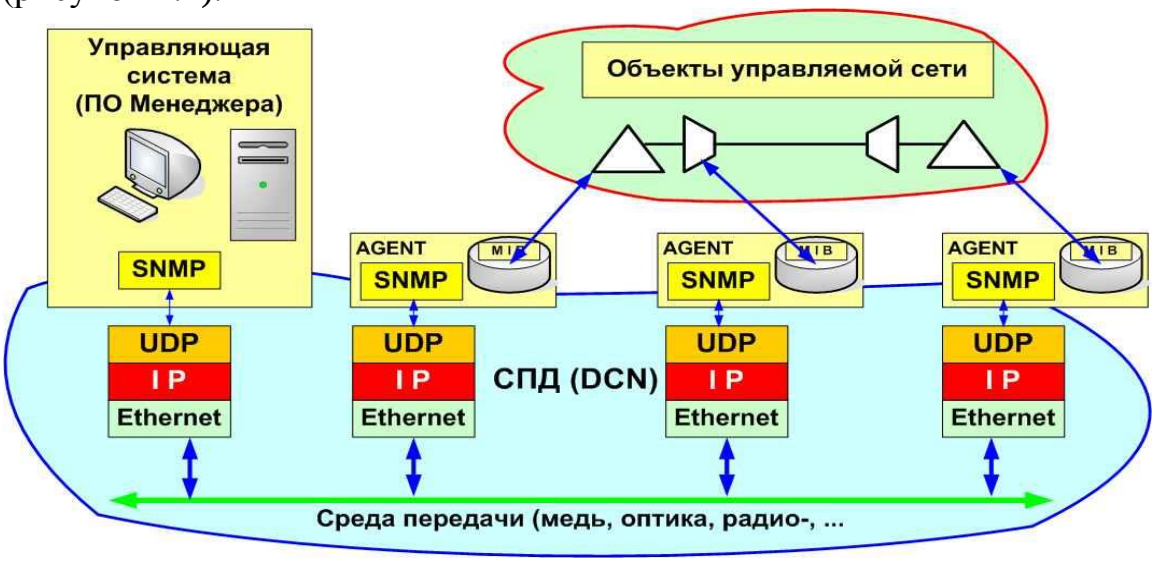


Рисунок 2.1 – Модель обмена управляющей информацией

Протокол SNMP стандартизован IETF в RFC-1157 (первая версия) и нашел широкое применение, как относительно простой, дешевый и в тоже время достаточно функциональный, т.е. позволяющий менеджеру производить опрос баз данных управляющей информации (Management Information Base – MIB), распределенных по управляемым объектам (обычно MIB входит в состав программного обеспечения агентов). Протокол SNMP также позволяет агенту извещать менеджера в случае незапланированных событий на стороне управляемого объекта (обычно в случае так называемых «алармов» - тревог).

В данном пособии рассматриваются основные функции протокола SNMP, состав и формат управляющих сообщений протокола, примеры кодирования этих сообщений, принципы обмена управляющей информацией с помощью сообщений данного протокола.

Сообщения управляющего протокола SNMP, которыми обмениваются между собой менеджер и агенты, вкладываются в информационную часть протоколов UDP/IP. Для доставки этих сообщений могут использоваться транспортные протоколы ATM, Ethernet, PPP и др.

Требуемая пропускная способность сетевых интерфейсов в части управляющей информации определяется размерами управляемой сети, количеством агентов, объемом необходимой управляющей информации и т.п. На практике, администратор сети для пропуска управляющего трафика предусматривает избыточность около 5% от пропускной способности конкретного интерфейса.

2.1.1 Функции менеджера и агента при обмене управляющей информацией

Сеть управления может быть построена как иерархическая, так и одноуровневая (одноранговая). Количество менеджеров в сети управления одного оператора может быть несколько, и зависит от многих факторов, например, для гетерогенной сети, построенной на оборудовании разных производителей, оператор вынужден использовать несколько управляющих систем, т.к. интегрированные системы управления либо очень дороги, либо ограничены по количеству и типам поддерживаемых MIB (баз данных управляющей информации).

В данном пособии рассматриваем вариант сети управления одноранговый с одним менеджером и несколькими, территориально распределенными (по управляемым объектам) агентам.

В центре управления сетью устанавливается сервер **с программным обеспечением менеджера**, выполняющим основные функции сбора, хранения, обработки, анализа управляющей информации, а также принятия управляющих решений. Для работы с персоналом центра управления, менеджер поддерживает функции интерфейсы F, G, обеспечивая работу нескольких рабочих мест (WS или АРМ). С этих рабочих мест персонал может выполнять администрирование сети.

Управляемое устройство, на котором функционирует **программа-агент**, может быть любого типа — например, сервер доступа в Интернет, УПАТС, АТС, мультиплексор SDH, маршрутизаторы, концентраторы и т.п. Программы управления должны быть построены таким образом, чтобы минимизировать воздействие программы-агента на управляемое устройство.

Программы-агенты по заданию менеджера или автоматически могут отслеживать следующие показатели работы сетевого оборудования:

- число и состояние каналов;
- сообщения о неисправности (сигналы тревог – alarm);
- число байтов и пакетов, входящих и исходящих из управляемого устройства;
- длина очереди в буферной памяти управляемого маршрутизатора;
- пропускная способность управляемого интерфейса и т.д.

В целом, функции управления определены рекомендацией ITU-T X.700, в виде так называемых **5-ти функциональных областей**:

- управление конфигурацией сети – **CM (Configuration Management)**,
- устранение последствий отказов – **FM (Faults Management)**,
- управление рабочими характеристиками – **PM (Performance Management)**,

- управление расчетами – **AM (Account Management, Billing)**,
- управление защитой информации (безопасностью) – **SM (Security Management)**.

В ряде рекомендаций серии X.73x, X.74x, X.800 и M.3xxx эти функциональные области детализируются до отдельных функций и процедур.

2.2 Схема взаимодействия менеджера и агента

В данной части проекта необходимо отразить структурную схему Вашего варианта сети, включая менеджера, часть сетевых узлов и агентов при этих узлах.

За основу можно взять схему рис.2.1, а также типовые схемы мультисервисных транспортных сетей и сетей доступа.

2.3 Стеки протоколов для обмена управляющей информацией

Интерфейс между менеджером и агентами реализуется посредством стека сетевых и прикладных протоколов. В данном пособии рассмотрим вариант обмена управляющей информации по стеку протоколов **SNMP/UDP/IP/Ethernret**, часто используемому для управления локальными сетями.

2.3.1 Стек протоколов IETF (TCP-UDP/IP)

На рисунке 2.2 иллюстрируется данный стек протоколов в сравнении с семиуровневой моделью ISO.

Уровни OSI-ISO			Уровни стека IETF
Приложения			Приложения (HP_OV, TNG, …)
7 Прикладной 6 Представительный 5 Сеансовый		Уровень сервисных протоколов	→ SNMP (Порты 161,162)
4 Транспортный		Транспортный	→ TCP, UDP
3 Сетевой		Сетевой (WAN)	→ IP
2 Канальный 1 Физический		Уровень сетевых интерфейсов (LAN, MAN)	→ Ethernet, ATM, FR, LAPD, …

Рисунок 2.2 – Стек протоколов SNMP/UDP/IP/Ethernet

В данной части проекта необходимо отразить профили протоколов по маршруту от менеджера до любого агента, включая стеки протоколов по узлам транспортной сети.

2.3.2 Форматы заголовков протоколов стека SNMP/UDP/IP/Ethernet.

Для передачи SNMP-сообщений используются протоколы UDP/IP/Ethernet, вносящие некоторую избыточность за счет своих заголовков.

На рисунке 2.3 приводятся общие форматы заголовков протоколов, входящих в этот стек.

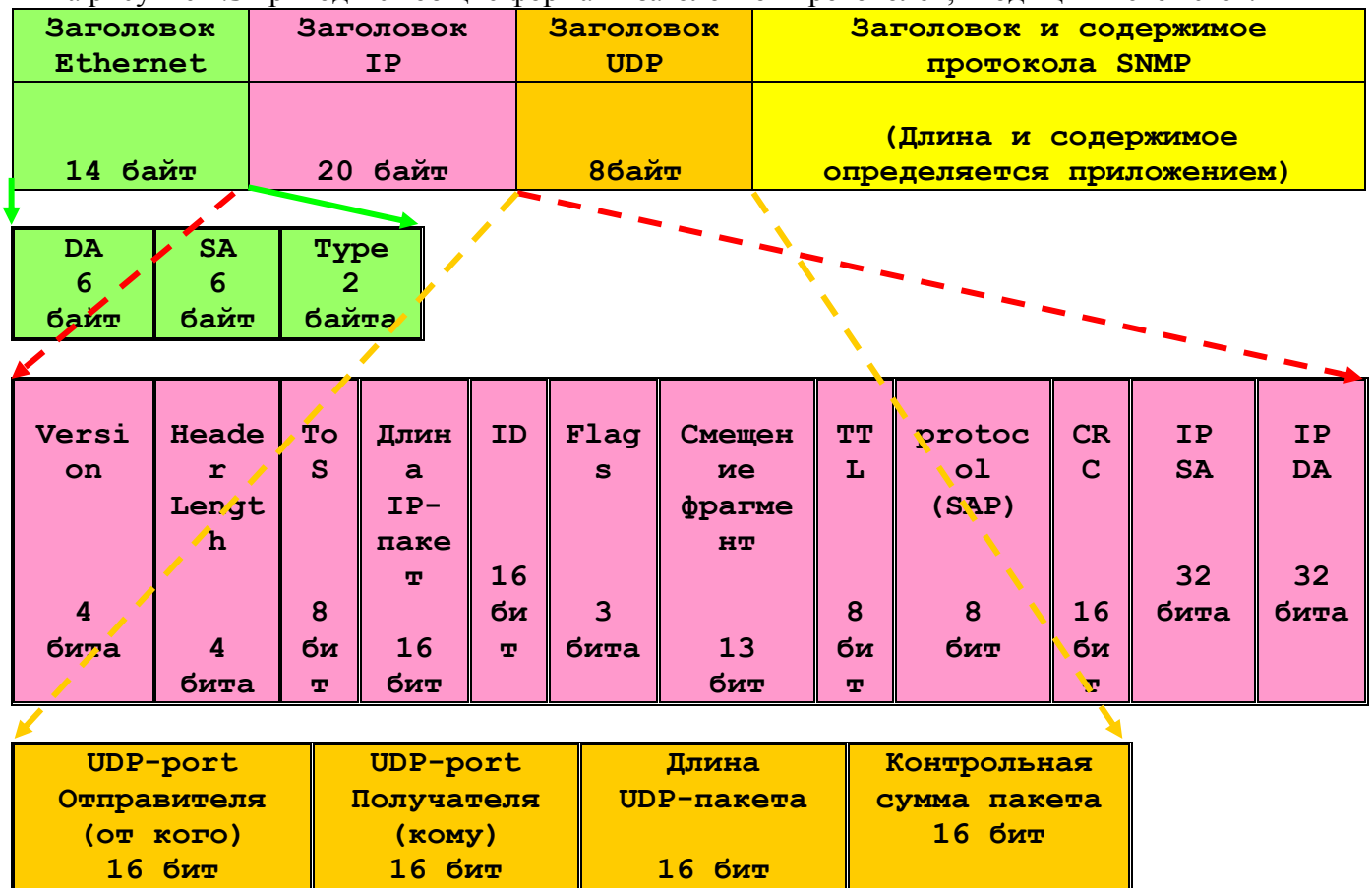


Рисунок 2.3 – Форматы заголовков протоколов Ethernet/IP/UDP

Поля протокола SNMP кодируются в соответствии с правилами BER (X.209) и в общем случае представлены в виде формата T-L-V (Тэг-Длина-Значение).

Содержимое этих полей зависит от конкретного типа PDU (протокольного блока данных).

В первой версии протокола SNMP (SNMPv1) было предложено пять типов PDU (**Get, Get-next, Set, Response и Trap**), форматы которых несколько отличаются между собой.

Более подробно форматы заголовков и содержимого протокола SNMP рассматривается в следующем разделе и составляет основную часть предлагаемого задания.

Конкретные трассировки сообщений протокола SNMP, измеренные у реального Интернет-провайдера приводятся в вариантах заданий. Все трассировки сообщений там представлены в шестнадцатиричных кодах и задачей студента является расшифровать значение полей этих сообщений по предлагаемой методике.

Общий формат сообщений SNMP. Представление о формуле T-L-V – смотри в описании языка ASN.1 и правил кодирования BER.

Version (INTEGER)	Community (OCTET STRING)	SNMP-PDUs (смотри ниже)
T-L-V	T-L-V	T – L- V

Рисунок 2.4 – Общий формат сообщений протокола SNMP

Форматы основных протокольных блоков (PDU-SNMP) представлены ниже.

Формат PDUs SNMP Get, Get-next, Set и Response (T-L-V)

Request ID (get, set, response)	Error status	Error index	Variable bindings
T – L- V	T-L-V	T-L-V	T – L- V

Формат PDU SNMP Trap (T-L-V)

Enterprise	Agent Address	Generic Trap Type	Specific Trap code	Time stamp	Variable bindings
T – L- V	T-L-V	T-L-V	T-L-V	T-L-V	T – L- V

Рисунок 2.5 – Форматы сообщений протокола SNMP

2.4 Основы управляющего протокола SNMP

Назначение и функции протокола

Протокол SNMP был разработан с целью **проверки** функционирования сетевых маршрутизаторов и мостов.

В системах управления на основе протокола SNMP, стандартизуются следующие элементы:

- протокол взаимодействия агента и менеджера, т.е. SNMP;
- язык описания моделей MIB и сообщений SNMP — язык абстрактной синтаксической нотации ASN.1 (стандарт ISO 8824:1987, рекомендации ITU-T X.208);
- несколько конкретных моделей MIB (MIB-I, MIB-II, RMON, RMON 2), имена объектов которых регистрируются в дереве стандартов ISO.

Сегодня протокол SNMP используется при управлении любыми видами оборудования и ПО в телекоммуникациях. Агенты SNMP встраиваются в аналоговые модемы, модемы ADSL, коммутаторы ATM и т. д.

SNMP — протокол прикладного уровня в стеке TCP/IP. SNMP используется для получения от сетевых устройств информации об их статусе, производительности и других характеристиках, которые хранятся в базе данных управляющей информации MIB (Management Information Base).

Простота SNMP определяется простотой MIB SNMP, особенно их первых версий MIB I и MIB II. Сам протокол SNMP также несложен.

Основные операции по управлению вынесены в ПО **менеджера.**

Для того, чтобы объект был виден для управления со стороны менеджера, необходимо внедрить на управляемом объекте ПО **агента** с поддержкой протокола SNMP и соответствующей базы данных управляющей информации – MIB.

Агент в протоколе SNMP — это обрабатывающий элемент, который выполняет пассивную роль, передавая в ПО менеджера по его запросу значения накопленных статистических переменных, тем самым обеспечивая доступ к значениям переменных MIB и дает менеджеру возможность реализовывать функции по управлению и наблюдению за устройством.

При этом устройство, имеющее встроенного агента должно работть с минимальными издержками на поддержку протокола SNMP, а основная производительность устройства с этим агентом должна быть использована для выполнения своих основных функций маршрутизатора, моста или концентратора, а агент занимается сбором статистики и значений переменных состояния устройства и передачей их менеджеру системы управления.

Протокол SNMP допускает возможность не только проверки, но и **внесения изменений** в функционирование указанных устройств.

Вся информация об объектах системы-агента подержится в так называемой **MIB (management information base) – базе управляющей информации**, другими словами MIB представляет собой совокупность данных об объектах, доступных для операций записи-чтения для конкретного менеджера (см. раздел 2.3.3 – Базы данных управляющей информации – MIB).

Есть стандарты, определяющие структуру MIB, в том числе набор типов ее объектов, их имена и допустимые операции над этими объектами. Древовидная структура MIB стандартизована ISO и ITU-T и содержит обязательные (стандартные) поддеревья, а также частные (private) поддеревья, позволяющие изготовителю сетевых устройств управлять специфическими функциями на основе стандартизованных объектов MIB.

Собственно функции протокола SNMP, реализуемые посредством соответствующих сообщений, сводятся к следующим:

- **Опросить содержимое MIB на стороне агента**
- **Изменить состояние переменных в MIB агента**
- **Ответить на запросы менеджера**
- **Уведомить менеджера о нештатных ситуациях на стороне агента**

Версии протокола SNMP

Первая версия этого протокола была опубликована организацией IETF в 1990-м году в документе – RFC-1157 и ориентировалась на управление **в локальных сетях.**

С развитием сети Интернет протокол SNMP стал использоваться для управления территориально удаленными объектами, и тогда выявились такие недостатки первой версии SNMPv1 как:

- незащищенность от несанкционированного доступа,
- избыточность и
- недостаточная функциональность.

В последующем недостатки первой версии постепенно ликвидировались в версиях протокола

- SNMPv2 (RFC-1901…1910) и
- SNMPv3 (RFC-3410…3419).

При этом название «Simple - Простой» для последующих протоколов после первой версии уже не совсем соответствует действительности.

Одновременно для поддержки новых функций в следующих версиях SNMP, разрабатывалист соответствующие базы данных управляющей информации (MIB), ориентированные на версии SNMPv2, SNMPv3.

Более подробно о дополнительных функциях SNMP, появившихся в новых версиях, а также о новых версиях MIB будет описано ниже.

Сфера действия протокола SNMP в настоящее время включает любые сетевые устройства, такие как хабы, шлюзы, хосты и т.д.

Недостатки протокола SNMP

Указанные ниже недостатки имеют отношение в основном к первой версии протокола (RFC-1157), разработанной для управления локальными сетями, где безопасность и надежность поддерживаются за счет ограниченных размеров управляемой сети, а такой недостаток как информационная избыточность — не является решаюшим в локальной сети.

Тем не менее, при использовании протокола SNMPv1 за пределами локальной сети, необходимо осознавать его недостатки (впрочем, хорошо известные хакерам):

- **Низкая безопасность**. Отсутствие средств взаимной аутентификации агентов и менеджеров. Единственное средство идентификации — «строка сообщества» — «community string». Эта строка в сообщении SNMP передаётся в открытой форме и служит основой для деления агентов и менеджеров на «сообщества», так что агент взаимодействует только с теми менеджерами, которые указывают в поле community string ту же символьную строку, что и строка, хранящаяся в памяти агента. По сути это не способ аутентификации, а способ структурирования агентов и менеджеров.

- **Низкая надежность.** Работа через ненадёжный протокол UDP (подавляющее большинство реализации агентов SNMP) приводит к потерям аварийных сообщений (сообщений trap) от агентов к менеджерам, что может привести к некачественному управлению.
- **Высокая информационная избыточность.** SNMP-Агенты не отличаются особым интеллектом, в частности не могут производить на месте какую-либо серьёзную обработку запросов от менеджеров. Вместо этого SNMP-агент транслирует всю информацию к менеджеру, где и происходит основная обработка. Это часто создает избыточный служебный трафик. В локальной сети этого можно не замечать, но если управление удаленными объектами ведется через арендованные каналы сетей общего пользования, то избыточный трафик сопровождается также издержками на его оплату.

Разработчики платформ управления стараются преодолеть эти недостатки. Например, в платформе HP OpenView Telecom DM TMN, являющейся платформой для разработки многоуровневых систем управления в соответствии со стандартами TMN и ISO, работает новая версия SNMP, организующая надежный обмен сообщениями между агентами и менеджерами за счет самостоятельной организации повторных передач сообщений SNMP при их потерях.

Сообщения (примитивы) протокола SNMP.

В SNMP агент взаимодействует с менеджером по принципу запрос-ответ. Генерируя какой-либо запрос, менеджер тем самым осуществляет определенное **действие (операцию)** по управлению объектом.

Действия менеджера и агента реализуются посредством обмена специфицированными протокольными блоками данных – **PDU-SNMP**, обозначаемых как запросы (команды) и ответы.

В данных методических указаниях мы будем считать синонимами термины **SNMP-сообщение** и **протокольный блок данных - PDU**.

Агент по своей инициативе генерирует только одно действие, называемое ловушкой ("trap" - ловушка). Другие действия агентов сводятся к ответам (Response) на запросы менеджера.

Менеджеры могут генерировать **три** (в версии SNMPv1) вида запросов – GetRequest, GetNextRequest, SetRequest.

Итак, всего в версии SNMPv1 определены 5 типов запросов-ответов (PDU).

- **GetRequest** – Этот PDU реализует функцию опроса управляемых объектов. Позволяет получить от агента содержимое одного объекта из MIB. Часто используется сокращенная запись – **Get**.
- **GetNextRequest** – Этот PDU реализует операцию получения следующего экземпляра из **таблицы**. С помощью этой операции просматривается весь список объектов. Как только встречается **первый объект из другой ветки MIB**, операция **прекращается**. Сокращенное название звучит как **GetNext**.
- **SetRequest** – Этот PDU реализует реализует функцию изменения данных в MIB. Часто о ней говорят просто как о команде **Set**. С помощью команды Set происходит собственно управление устройством. Обычно определяются реакции агента на такие события, как инициализация агента, рестарт агента, обрыв связи, восстановление связи, неверная аутентификация и потеря ближайшего маршрутизатора. Если происходит любое из этих событий, то агент инициализирует прерывание.
- **GetResponse** – Этот PDU выполняется агентом в ответ на команды GetRequest, GetNextRequest, SetRequest. Если это ответ на первые две, то внутрь будут вложены запрошенные данные, если это ответ на Set, то внутри будет подтверждение об успешном выполнении операции Set. Самые распространенные названия для этой команды **Reply** или **Response**.
- **Trap** – Этот PDU позволяет агенту реализовать функцию оповещения менеджера о том, что на управляемом объекте произошла нештатная ситуация (тревога – alarm). Содержит внутри себя специальный идентификатор объекта – **OID**, показывающий, что это ловушка, а также информацию о том какой MIB-объект установил ловушку и данные этого объекта.

В общем виде процедуры обмена информацией между менеджером и агентом можно сформулировать следующим образом (см. рис. 2.6):

1. Менеджер формирует и посылает агенту стандартное сообщение-запрос для получения информации об объекте, на котором расположен данный агент;
2. Агент формирует ответ на запрос и посылает этот ответ менджеру;
3. Менеджер на основе полученной информации о состоянии объекта формирует и посылает агенту стандартное сообщение об изменении параметров;
4. Агент, получив сообщение на изменение параметров в MIB, посылает сообщение-прерывание и производит соответствующую реконфигурацию.

Рисунок 2.6 – Процедуры протокола SNMP

Сообщения, содержащие такие блоки данных протокола как GetRequest-PDU, GetNextRequest-PDU, SetRequest-PDU, отправляются от менеджера к агенту, а сообщения, содержащие GetResponse-PDU и Trap-PDU, отправляются от агента к менеджеру.

Менеджер отправляет свои три запроса на UDP порт 161. Агент отправляет «ловушки» (trap) на UDP порт 162. Так как используются два разных порта, одна система может выступать в роли менеджера и агента одновременно.

Состав и формат сообщений протокола SNMP можно привести в традиционной форме, отображающей различные поля заголовков и информационной части.

На рис.2.7 представлен традиционный формат для SNMP-сообщений Get, GetNext Set и Response.

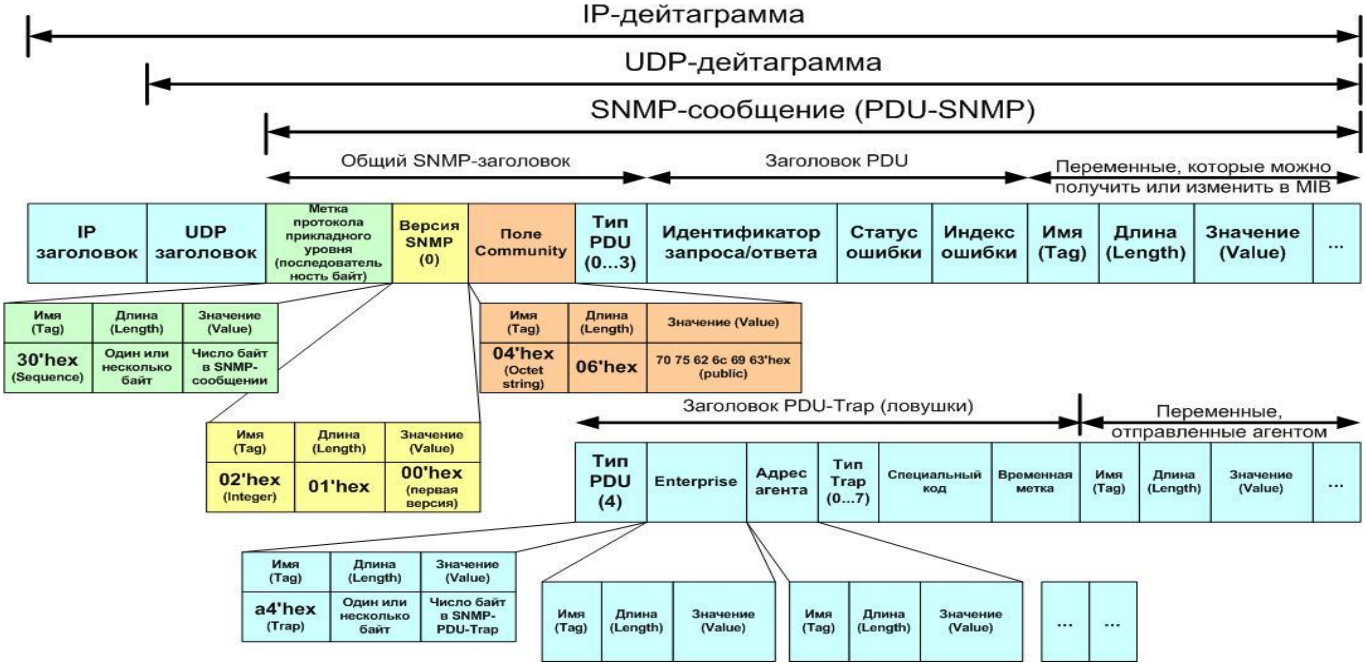


Рисунок 2.7 – Структура сообщения SNMP

Каждая часть сообщения SNMP кодируется в формате T-L-V (Тэг-Длина-Значение), в соответствии с правилами кодирования BER, описанными в рекомендации ITU-T X.209.

С учетом правил кодирования приведем формат сообщений протокола SNMP в следующем виде (см. рис. 2.8):

Заголовок SNMP									**Заголовок PDU** (для Get, GetNext, Set, Response)									**Переменные** (Идентификаторы объектов в MIB, значения параметров)								
Версия			Пароль (Community)			Тип PDU			Идентификатор PDU			Статус ошибки			Индекс ошибки			Имя (Tag)	Длина (L)	Значение (Value)					
Vers – 1			По умолчанию: **public**			См. табл.1			Целое значение от 0 до 2^{32}-1			См. табл.2			См. ниже			***	****	Имя (Tag)	Длина (L)	…			
T	**L**	**V**	**T**	**L**	**V**	**T**	**L**	**V**	**T**	**L**	**V**	**T**	**L**	**V**	**T**	**L**	**V**	**T**	**L**	**T**	**L**	**V**			

Рисунок 2.8 – Формат SNMP-сообщений, вкладываемых в UDP-дейтаграммы

Каждое сообщение SNMP протокола начинается с заголовка, определяющего общие функции данного сообщения (спросить, изменить, ответить, уведомить).

В заголовок **любого SNMP-сообщения** входят следующие поля:

- **Версия протокола.** В этом поле указывается целое число на 1 меньше версии протокола (Version – 1, т.е. **номер версии SNMP минус один**).
- **Пароль доступа к управляемым ресурсам**. В качестве такого пароля в SNMPv1 используется поле Community, которое содержит строку байт (октетов). Если администратором не используется другое, то по умолчанию эти октеты кодируют символы "**public**", что означает общий доступ.
- **Тип PDU (Protocol Data Unit – Протокольный блок данных)**. Определяет код команды (запроса) или ответа и, соответственно, **основные функции данного сообщения** (спросить, изменить, ответить, известить и т.п.). Коды запросов приведены в табл.1.

Таблица 1 – Коды заголовков сообщений протокола SNMP

Сообщение SNMP/ Tag'bin	Тип PDU		Назначение PDU
	код 'dec	Tag 'hex	
Сообщения протокола SNMP первой версии			
GET-request/ `10 1 00000`	**0**	**a0**	Получить значение указанной переменной или информацию о состоянии сетевого элемента;
GET_next_request/ `10 1 00001`	**1**	**a1**	Получить значение переменной, не зная точного ее имени (следующий логический идентификатор на дереве MIB);
GET response/ `10 1 00010`	**2**	**a2**	Отклик на GET-request, GET_next_request и SET-request. Содержит также информацию о состоянии (коды ошибок и др);
SET-request/ `10 1 00011`	**3**	**a3**	Присвоить переменной соответствующее значение. Используется для описания действия, которое должно быть выполнено;
TRAP/ `10 1 00100`	**4**	**a4**	Отклик сетевого агента на событие или на изменение состояния.
В последующих версиях SNMP (2 и 3) функции протокола были значительно расширены, для чего было добавлено несколько новых сообщений (см. ниже).			
GetBulkRequest/ (v2)	**5**	**a5**	Запрос пересылки больших объемов данных, например, таблиц.
InformRequest/ (v2)	**6**	**a6**	Менеджер обращает внимание партнера на определенную информацию в MIB.
SNMPv3-Trap/ (v3)	**7**	**a7**	Отклик на событие (расширение по отношению v1 и v2).
Report (v3)	**8**		Отчет (функция пока не задана).

В следующей части сообщения SNMP содержится **заголовок** конкретного **PDU**, причем для PDU типа Get, GetNext, Set, Response формат этого заголовка одинаков, а для сообщений **Trap** формат заголовка несколько иной (см. рис. 2.9).

Итак, рассмотрим элементы заголовков PDU типа Get, GetNext, Set, Response:

- **Идентификатор PDU.** Так как менеджер генерирует множество запросов, то для их идентификации используется данное поле. С помощью идентификатора запрсы и ответы на них связываются в пары (т.е. запрос и ответна данный запрос – имеют одинаковый идентивикатор). Может принимать значения от 0 до 2^{32}-1. Для запросов Get, GetNext и SET значение **идентификатора запроса** устанавливается менеджером и возвращается агентом объекта управления в отклике Response, что и позволяет связывать в пары запросы и ответы.
- **Статус ошибки.** Поле **статус ошибки** характеризуется целым числом (код ошибки), присланным объектом управления.
- **Индекс ошибки.** Если произошла ошибка, поле **индекс ошибки (error index)** характеризует, к какой из переменных это относится. Значение **error index** является указателем переменной и устанавливается агентом объекта управления не равным нулю для ошибок типа **badvalue**, **readonly** и **nosuchname**.

Таблица 2 - Коды ошибок

Статус ошибки	Имя ошибки	Описание
0	Noerror	Все в порядке;
1	Toobig	Объект не может уложить отклик в одно сообщение;
2	Nosuchname	В операции указана неизвестная переменная;
3	badvalue	В команде set использована недопустимая величина или неправильный синтаксис;
4	Readonly	Менеджер попытался изменить константу;
5	Generr	Прочие ошибки.

После указанных заголовков, следует **информационная часть сообщения SNMP**, в которой могут размещаться одна или более переменных, составляющих собственно управляющую информацию, запрашиваемую менеджером.

Кодирование всех элементов сообщений протокола SNMP, производится согласно правилам кодирования, специфицированным в рек. ITU-T X.209 (**BER** – Basic Encoding Rules – базовые правила кодирования). В разделе 2.3 приводятся основные сведения об этих правилах.

Важно понимать, что каждый элемент сообщения, включая, те элементы, значения которых равно 0, кодируется **как минимум тремя байтами** по структуре **T-L-V** (Tag-Length-Value).

Для расшифровки полей пакета, относящихся к протоколу SNMP, необходимо познакомиться с основами **языка ASN.1**, который используется для описания информационных элементов (ИЭ), входящих в сообщения протокола SNMP, а также с правилами кодирования (BER), используемыми для кодирования сообщений протокола и ИЭ. Все сообщения протокола SNMP относятся к контекстно-зависимому классу тэгов (class tag=10) и имеют составной вид информационных элементов (constructor).

Для сообщений Trap, формат сообщения несколько отличается в части **заголовка PDU**:

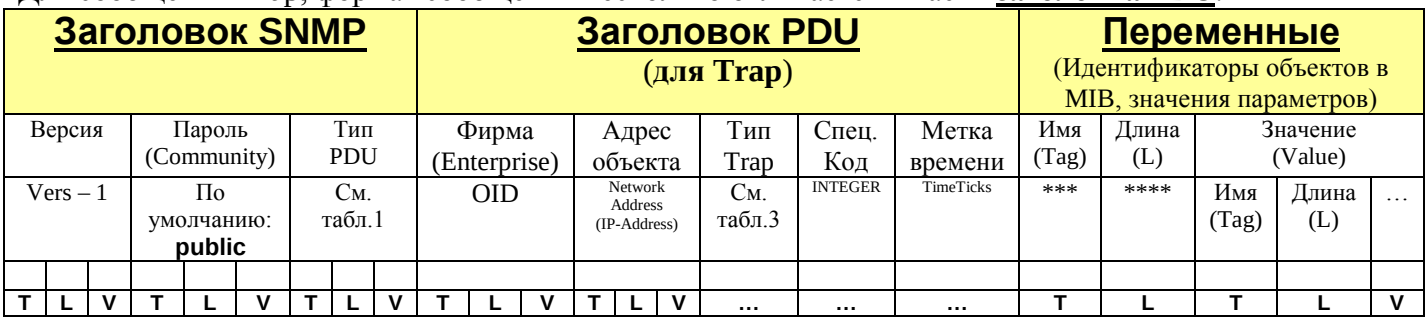


Рисунок 2. 9 – Формат PDU-SNMP-Trap

Поясним назначение полей заголовка PDU-Trap:
Поле **фирма** (**enterprise**) – характиризует производителя опрашиваемого объекта;
В качестве **адреса агента** используются сетевыой адрес объекта, на котором установлень агент, в частности IP-адрес;
Тип ловушки (Trap) является целым числом, кодирующим один из следующих типов (см. табл.2):
Таблица 3 – Коды TRAP

Код TRAP	Тип TRAP	Описание
0	Coldstart	Установка начального состояния объекта.
1	Warmstart	Восстановление начального состояния объекта.
2	Linkdown	Интерфейс выключился. Первая переменная в сообщении идентифицирует интерфейс.
3	Linkup	Интерфейс включился. Первая переменная в сообщении идентифицирует интерфейс.
4	Authenticationfailure	От менеджера получено SNMP-сообщение с неверным паролем (community).
5	EGPneighborloss	EGP-партнер отключился. Первая переменная в сообщении определяет IP-адрес соседа.
6	Entrprisespecific	Информация о TRAP содержится в поле специальный код.

Для кодов TRAP 0…4 поле **специальный код** должно быть равно нулю.
Поле **временная метка** содержит число сотых долей секунды (число тиков) с момента инициализации объекта управления. Так прерывание coldstart выдается объектом через 200 мс после инициализации.

Способы кодирования сообщений протокола SNMP

Язык описания информационных элементов (объектов) – ASN.1

Информационная модель Системы управления представляется одной из самых сложных моделей и содержит огромное множество управляемых объектов, их атрибутов (свойств), действий (операций), реакций на действия и т.п. элементов.

В терминах информационной модели это множество объектов, отражающих различные свойства многообразных ресурсов, представлено **абстрактным множеством информационных элементов (ИЭ)**, которые имеют **конкретные значения**, т.е. отличающиеся друг от друга **элементы** этого множества.

Значения (элементы) разделяются на типы, представляющие некоторое **подмножество значений**, которому присвоено **имя**.

МККТТ, учитывая:

- многообразие и сложность информационных объектов на прикладном уровне;
- необходимость нотации (записей о свойствах объектов) высокого уровня для абстрактного описания таких объектов;
- преимущества от выделения и стандартизации правил кодирования таких информационных объектов

рекомендует нотацию для определения абстрактного синтаксиса информационных объектов – **ASN.1** (X.208), а также определяет типы и подтипы информационных объектов и правила кодирования – **BER** (Basic Encoding Rules - X.209).

Нотация ASN.1 широко используется при описании многих стандартов OSI, в частности моделей управляемых объектов, структуры сообщений протокола CMIP, OMAP и SNMP, переменных MIB, а также в качестве нотации для описания терминов информационных протоколов верхних уровней (например, FTP, MAP, INAP и т.п.).

Нотация ASN.1 служит для установления однозначного соответствия между терминами, взятыми из стандартов, предназначенных для использования человеком, и теми данными, которые передаются в коммуникационных протоколах.

Достигаемая однозначность очень важна для гетерогенной среды, характерной для современных сетей. Так, вместо того чтобы указать, что некоторая переменная протокола

представляет собой определенное число, разработчик протокола, использующий нотацию ASN.1, должен определить формат и допустимый диапазон переменной. В результате документация на MIB, написанная с помощью нотации ASN.1, может механически транслироваться в форму кодов, характерных для сообщений протоколов верхних уровней.

Нотация ASN.1 похожа на другие метаязыки, используемые при описании языков программирования, в частности C++.

Нотация ASN.1 поддерживает базовый набор различных типов данных, таких как целое число, строка и т. п., а также позволяет конструировать из этих базовых типов составные данные — массивы, списки, структуры.

В ASN.1 типы и значения выражаются в нотации, близкой к используемой в языках программирования. Идентификаторы объектов (имена значений и полей) и имена типов состоят из букв, цифр и пробелов. Идентификаторы начинаются со строчной буквы, а имена типов - с прописной.

В ASN.1 используются следующие символы:

Символы от **A** до **Z**
Символы от **a** до **z**
Символы от **0** tдо **9**
Символы **: = , { } < .**
Символы **( ) [ ] - ' ”**

В ASN.1 зарезервированы следующие последовательности символов (служебные слова)

BOOLEAN	OPTIONAL	INCLUDES
INTEGER	DEFAULT	**MIN**
BIT	COMPONENTS	**MAX**
STRING	UNIVERSAL	**SIZE**
OCTET	APPLICATION	**FROM**
NULL	PRIVATE	**WITH**
SEQUENCE	**TRUE**	COMPONENT
OF	**FALSE**	PRESENT
SET	**BEGIN**	ABSENT
IMPLICIT	**END**	DEFINED
CHOICE	DEFINITIONS	BY
ANY	EXPLICIT	PLUS-INFINITY
EXTERNAL	ENUMERATED	MINUS-INFINITY
OBJECT	EXPORTS	TAGS
IDENTIFIER	IMPORTS	

Существуют правила трансляции структур данных, описанных на ASN.1, в структуры данных языков программирования, например C++. Соответственно, имеются трансляторы, выполняющие эту работу.

ASN.1 описывает несколько способов описания типов данных (ИЭ). Прежде всего, это использование **простых и составных типов данных**.

ASN.1 различает следующие типы ИЭ:

- **Простой тип** (тип-примитив) – определяется прямым заданием (описанием) множества составляющих его значений;
- **Структурированный** (составной) тип (тип-конструктор) – это тип, при определении которого используются ссылки на другие типы. Фактически, структурированный тип – это своеобразная «матрешка», содержащая внутри себя ИЭ как простого типа, так и составного типа, что придает такой конструкции высокую гибгость, при описании большого числа взаимосвязанных переменных (например, баз данных управляющей информации – MIB). Число вложений в «матрешку» не ограничено, как не ограничено число ветвей дерева MIT.

В ASN.1 определено **несколько структурированных типов** ИЭ, например:

SEQUENCE	Упорядоченная последовательность из одного или более других типов ИЭ.
SET	Неупорядоченный набор из одного или более других типов ИЭ.
CHOICE	Набор из заданного списка возможных типов ИЭ
SEQUENCE OF	Одномерный массив ИЭ одного типа

Каждому типу в ASN.1 присвоено **обозначение**, выраженное в виде **ТЭГА (англ. –** TAG**, русские синонимы – указатель, индикатор, метка, описатель).**

ASN.1 определяет 4 класса тэгов (описателей).

- **1 класс** – Универсальный (**UNIVERSAL** - **UNI**) – используется в ASN.1 (X.208) и присваивается либо одному типу данных, либо способу построения типов.
- **2 класс** – Прикладной (общеприкладной – **APPLICATION-WIDE** – **APP-W**) – присваивается типам данных, определенных в других стандартах или Рекомендациях (например, в рек. X.500 для службы каталогов).
- **3 класс** – Контекстно-зависимый (**CONTEXT-SPECIFIC – C-SPEC**) – эти тэги могут назначаться любым типам данных и интерпретируются в соответствии с контекстом, в котором они используются.
- **4 класс** – Пользовательский (частный – **PRIVATE** - **PRIV**) – присваивается типам данных, определенных различными организациями, а не стандартами ISO или Рекомендациями МККТТ.

Базовые правила кодирования информационных элементов – BER

В ASN.1 определены правила преобразования **значений переменных** (ИЭ) в последовательность байтов для последующего обмена по сети. Называются эти правила – **базовые правила кодирования (BER – см. рек. ITU-T X.209).**

Каждое передаваемое значение ИЭ кодируется **тремя полями** (так называемая конструкция **T-L-V=Tag-Length-Value**):

1. **Идентификатор ИЭ (Тэг)** - Определяет класс и тип ИЭ, а также указывает, является ли метод кодирования примитивным или составным.
2. **Длина поля данных** - определяет число октетов содержимого.
3. **Значение (Содержимое) поля данных (Value, Content)** – т.е. сущность или конкретное выражение значения ИЭ.

T	L	V
Тэг	**Длина**	**Значение (Содержимое)**

Таким образом, перед передачей содержимого любого ИЭ, включая значение «0», передается Тэг, кратко поясняющий тип данных в поле содержимое, и длина поля «Содержимое», выраженная в байтах и записанная в поле «Длина».

Дадим более детальную характеристику этим полям.

Идентификатор типа объекта (Тэг).

Поле Tag представляется **8-ми битовым кодом (см. рис.2-1)**, 2 старших бита (7-й и 8-й), которого указывают класс тэга (см. рис.2.10).

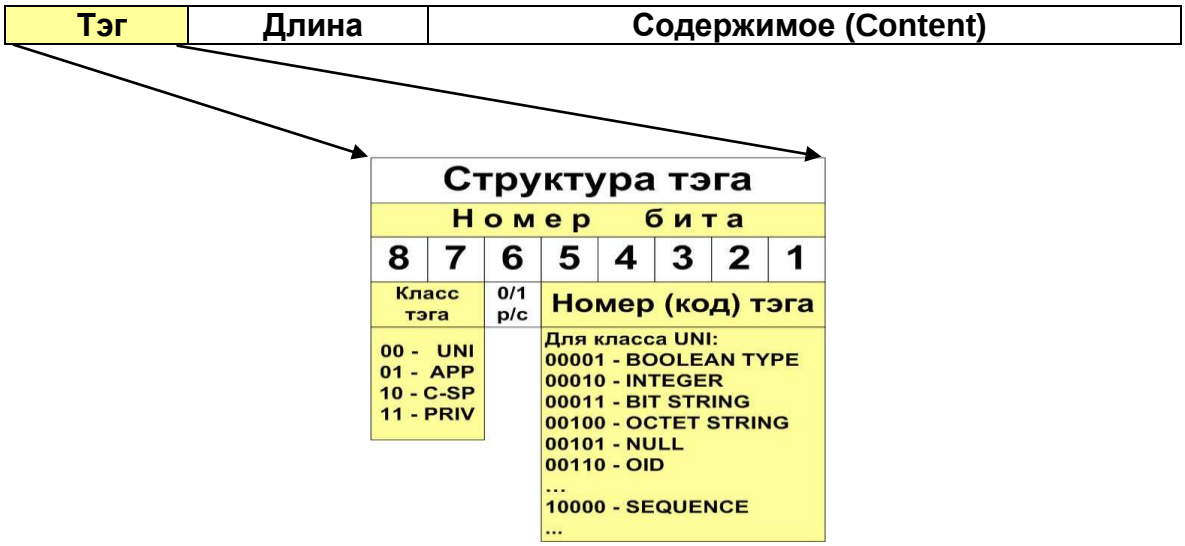


Рисунок 2.10 – Структура Тэга

Таблица 4 – Кодирование Класса тэгов (7-й и 8-й биты)

Класс		Бит 8	Бит 7
Универсальный	UNI	0	0
Прикладной	APP	0	1
Контекстно-ориентированный	C-SP	1	0
Частный	PRIV	1	1

6-й бит тэга указывает тип ИЭ. Возможно два типа данных ИЭ:

- **простой** тип данных ИЭ **(p - primitive)** – значение 6-го бита “0”,
- **составной** тип данных ИЭ **(c - constructor)** – значение 6-го бита “1”.

Для составных типов данных, характерна конструкция «**матрешки**»:

T – L - (T – L - (T – L (…V)))

Типичными значениями **составных данных** являются:

- последовательности данных (**Sequence**),
- наборы данных (**Set**),
- выбор из набора (ассортимента) данных (**Choice**).

Оставшиеся 5 бит тэга могут использоваться для кодирования значений (номера) тэга. Причем этих 5 бит достаточно для записи кода не превышающего значения 30 (11110’Bin). Если значения 5 бит заполняются “1”, то используется многобайтовая структура тэга, при этом реальное значение кода тэга указывается в следующем байте или байтах.

Для **универсального класса** (**UNI**) предусмотрено не больше 30 кодов тэгов, описанных в рек. ITU-T X.208 (см. табл. 5).

Таблица 5 – Назначенные коды тэгов класса **UNI**

Класс (биты 8 и 7)	р/с (бит 6)	Код (биты 54321)	Типы данных в поле **«содержимое»**	**Tag ‘hex**
UNI (00)	0	00001’Bin или 1’Dec	Boolean type	01
UNI (00)	**0**	**00010**’Bin или **2**’Dec	**Integer type**	**02**
UNI (00)	0	00011’Bin или 3’Dec	Bitstring type	03
UNI (00)	0	00100’Bin или 4’Dec	Octetstring type	04
UNI (00)	0	00101’Bin или **5**’Dec	**Null type**	05
UNI (00)	**0**	**00110**’Bin или **6**’Dec	**Object identifier type**	**06**
UNI (00)	0	7’Dec	Object descriptor type	07
UNI (00)	0	8’Dec	External type	08
UNI (00)	0	9’Dec	Real type	09
UNI (00)	0	10’Dec	Enumerated type	0a
UNI (00)		12…15’Dec	Reserved for future versions	
UNI (00)	**1**	**16’Dec**	**Sequence and Sequence-of types**	**30**
UNI (00)	1	17	**Set and Set-of types**	31
UNI (00)	0/1	18…22 25…27	Character string types (например, последовательность символов в коде **IA5**)	12/32
UNI (00)	0	23, 24	Time types	17, 18
UNI (00)		28…	Reserved for future versions	

Для **других классов тэгов** существуют свои способы кодирования, рассматриваемые в соответствующих документах.

Например, в рек. ITU-T Q.773 рассматривается как универсальный класс тэгов (00), так и **прикладной (01)** для протокола TCAP. Например, для таких информационных элементов, как тип сообщения протокола TCAP используются следующие номера тэгов из класса «Прикладной» (01) – см. табл.6.

Таблица 6 – кодирование типов сообщений в классе тэгов «Прикладной или 01» для протокола **TCAP** (Q.773 ITU-T)

	\номер бита в тэге	**8**	**7**	**6**	5	4	3	2	1
Тэг’hex	**Тип сообщения**	**Класс тэга**		**Тип**	Номер тэга				
61’hex	**Unidirectional**	**0**	**1**	1	0	0	0	0	1
62’hex	**Begin**	**0**	**1**	1	0	0	0	1	0
63’hex	(reserved)	**0**	**1**	1	0	0	0	1	1
64’hex	**End**	**0**	**1**	1	0	0	1	0	0
65’hex	**Continue**	**0**	**1**	1	0	0	1	0	1
66’hex	(reserved)	**0**	**1**	1	0	0	1	1	0
67’hex	**Abort**	**0**	**1**	1	0	0	1	1	1

В рек. ITU-T X.219, X.229 рассматриваются услуги протокола ROSE и соответствующие операции. Для кодирования этих информационных элементов (операций) используется класс тэгов **контекстно-зависимый** (**C-SP** или 10).

Пример кодирования этих операций приведен в табл. 7

Таблица 7 – Кодирование тэгов для операций протоколов **ROSE, TCAP**.

	\номер бита в тэге	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
Тэг’hex	**Тип операции**	**Класс тэга**		**Тип**	Номер тэга				
a1’hex	**Invoke**	**1**	**0**	**1**	0	0	0	0	1
a2’hex	**Return Result (Last)**	**1**	**0**	**1**	0	0	0	1	0
a3’hex	**Return Error**	**1**	**0**	**1**	0	0	0	1	1
a4’hex	**Reject**	**1**	**0**	**1**	0	0	1	0	0
a5’hex	(reserved)	**1**	**0**	**1**	0	0	1	0	1
a6’hex	(reserved)	**1**	**0**	**1**	0	0	1	1	0
a7’hex	**Return Result (Not Last)**	**1**	**0**	**1**	0	0	1	1	1

Подобный же класс тэгов (контекстно-зависимый или 10) используется для кодирования операций (команд, сообщений или PDU) в протоколе **SNMP** (стандарт IETF RFC-1155) – см. табл.8.

Таблица 8 – Кодирование тэгов для PDU протокола **SNMP**.

	\номер бита в тэге	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
Тэг'hex	**Тип операции (PDU)**	**Класс тэга**		**Тип**	Номер тэга				
a0'hex	**GET-request**	**1**	**0**	**1**	0	0	0	0	0
a1'hex	**GET_next_request**	**1**	**0**	**1**	0	0	0	0	1
a2'hex	**GET response**	**1**	**0**	**1**	0	0	0	1	0
a3'hex	**SET-request**	**1**	**0**	**1**	0	0	0	1	1
a4'hex	**TRAP**	**1**	**0**	**1**	0	0	1	0	0

Длина поля данных

Возможно два способа представления этого поля:

1. Краткая форма (см. рис.2.11). Используется, если длина поля данных (содержимого) не превышает 127 байт. В этом случае старший (8-й) бит поля «длина» имеет значение «0», а в остальных семи битах записывается реальная длина поля данных в байтах, причем младшим (наименее значащим битом - LSB), является 1-й бит, а старшим (наиболее значащим битом - MSB), является 7-ой бит.

	Номер бита							
	8	**7**	**6**	**5**	**4**	**3**	**2**	**1**
Значение бита	**0**	**MSB**						**LSB**
		Длина поля данных (в байтах)						

Рисунок 2.11 – Краткая форма поля «Длина»

Например, если содержимое имеет длину 38 байт, то значение поля «Длина», будет следующим:

L= 00100110'Bin (38'Dec)

2. Длинная форма (см. рис.2.12). Используется, если длина поля данных (содержимого) более 127 байт. В этом случае старший (8-й) бит поля «длина» имеет значение «1», а в остальных семи битах **первого байта** записывается целое число, равное количеству байт в поле «Длина» - 1. Реальная длина поля данных, выраженная в байтах, записывается в последующих байтах поля «Длина». При этом, младшим (наименее значащим битом - LSB), является 1-й бит последнего байта, а старшим (наиболее значащим битом - MSB), является 8-ой бит первого байта.

Номер бита							
8	**7**	**6**	**5**	**4**	**3**	**2**	**1**
1	**MSB**						**LSB**
	количество байт в поле «Длина» -1						
MSB							
…							
количество байт в поле «Содержимое»							
							LSB

Рисунок 2.12 – Длинная форма поля «Длина»

Пример 1: если содержимое имеет длину 219 байт, то значение поля «Длина», будет следующим:

1-й байт - 10000001'Bin – указывает длину поля «Длина» (1 байт)

2-й байт - 11011011'Bin – указывает длину поля «Содержимое» (219 байт)

Пример 2: если содержимое имеет длину 1347 байт, то значение поля «Длина», будет следующим:

1-й байт - 10000010'Bin – указывает длину поля «Длина» (2 байта)

2-й байт - 00000101'Bin } – эти байты указывают длину поля
3-й байт - 01000011'Bin } «Содержимое» (1347 байт)

Или в другом виде эту запись можно представить так:

```
1-й байт  2-й байт  3-й байт
   82        05        43     'Hex
10000010  00000101  01000011'Bin=1347'Dec
```

- длина поля «Содержимое» (1347 байт)

- длина поля «Длина» (следующие 2 байта)

Содержимое данных (Content)

В этом поле записывается содержимое передаваемых данных ИЭ, тип и структура которых определены в предшествующем тэге.

Например, BER-кодирование **значения INTEGER** (ИЭ, значение которых выражено целым числом является всегда примитивным. Октеты содержимого могут состоять из одного или нескольких байт. Первой передается старшая цифра содержимого (целого числа).

Если в поле данных содержится целое число (INTEGER) менее 127, то оно кодируется с помощью одного байта. Если целое число больше или равно 127, то оно кодируется более чем одним октетом, причем старший (значащий) байт передается первым

Целое число со значением нуль кодируется **одним октетом 00.**

Примеры BER-кодирования значений INTEGER представлены в таблице 9.

Таблица 9. Примеры BER-кодирования значений INTEGER

Значение целого ('Dec)	**BER-код ('Hex)**
0	02 01 00 - значение поля «Содержимое» - целое число, равное 0. - длина поля «Содержимое» (1 байт) - тэг, обозначающие, что в поле «Содержимое», передается целое число (код тэга – 00010 - INTEGER), простого типа (значение 6-го бита равно 0), а тэг относится к классу универсальный – UNI (биты 8 и 7 имеют значение 0)
59	02 01 3b
127	02 02 00 7f
128	02 02 00 80
256	02 02 01 00
3471	02 02 0D 8F
427735	02 03 06 86 D7
-128	02 01 80
-129	02 02 ff 7f

Информационный элемент NULL

Тип NULL обозначает нулевую величину, используемую в качестве **параметра алгоритмов**.

Кодирование для типа NULL является всегда примитивным, октеты содержимого пусты. BER-представление значения NULL может иметь одну из приведенных ниже форм (зависит от используемого представления октетов длины).

```
1.  05 00  -     (краткая форма)
2.  05 81 00    -     (полная форма, т.е. T-L-V)
```

Кодирование октетов конца содержимого.

Октеты конца содержимого (End-of-contents - EOC), кодируются универсальным тэгом, имеют простой тип, код тэга 00000, длину 0 байт, а содержимое - отсутствует:

Тэг	Длина	Содержимое
00	00	Отсутствует

Другие примеры кодирования поля «Содержимое»:

Строки бит (**Bitstring type**) передаются в том виде как они есть, однако, есть проблема в том, что поле длины указывает длину в **байтах**, а не в битах. Поэтому первый байт поля данных будет указывать **сколько бит последнего байта не используется**.

Например, если необходимо передать строку бит: **0 1001 1111** (длиной 9 бит), то после BER-кодирования, передаваемая последовательность выглядит так:

- **00000011** – тэг, определяет – **класс – UNI**, **простой** тип, **строка бит**
- 00000011 – поле длины (содержимое в 3 байта для передачи 9-ти бит!)
- **00000111** – первый байт содержимого указывает, что в последнем байте не используется 7 бит.
- **01001111** – первый байт данных (8 бит).
- **10000000** – еще один бит данных – остаток строки бит (семь последних бит не используется!),

Таким образом, вместо передачи девяти бит 0 1001 1111, при BER-кодировании будет передана последовательность из **пяти байт** (40 бит!) – **03 03 07 4F 80**’Hex!

Строки символов IA5

Тип IA5_String представляет любые последовательности IA5-символов (международный алфавит IA5 - эквивалентен ASCII). Длина строки может быть любой, включая нуль.

Этот тип используется для адресов электронной почты и неструктурированных имен.

BER-кодирование величины IA5_String может быть примитивным или структурированным. При примитивном кодировании октеты содержимого представляют собой символы IA5 в ASCII-кодах.

Рассмотрим примеры представления значения IA5-строки “cent@neic.nsk.su”.

Короткая форма октетов длины

```
          c  e  n  t  @  n  e  i  c  .  n  s  k  .  s  u
12 11 63 65 6e 74 40 6e 65 69 63 2e 6e 73 6b 2e 73 75
T  L                    V
```

Длинная форма октетов длины

```
             c  e  n  t  @  n  e  i  c  .  n  s  k  .  s  u
12 81 11 63 65 6e 74 40 6e 65 69 63 2e 6e 73 6b 2e 73 75
T     L                    V
```

Большинство текстовых редакторов, включая Word, содержат встроенную всставку символов, вызвав которую можно перевести 16-ти’-ричные представления символов в текстовые!

Среди приведенных в табл. 5 различных типов данных информационных элементов, **особое место** занимает тип данных под названием **«Идентификатор объекта»**, широко используемый для обозначения управляемых объектов.

BER-кодирование этого типа данных значительно отличается от других типов данных и для его понимания необходимо рассмотреть некоторые аспекты, посвященные структуре управляющей информации (раздел 2.3.3.) и базам данных управляющей информации (MIB – раздел 2.3.4).

Рассмотрим более подробно этот тип данных.

Идентификатор объекта (OID).

Тип **OBJECT IDENTIFIER – OID** используется для идентификации содержимого таких информационных элементов (ИЭ), как:

- алгоритмов в X.509,
- атрибутов Directory Information Base (DIB) в протоколах DAP (X.501), LDAP
- управляемых объектов в MIB и др.

BER-кодирование OBJECT IDENTIFIER является всегда **примитивным**.

Значения OBJECT IDENTIFIER присваиваются при регистрации информационного элемента соответствующей организацией.

Рассмотрим использование **OBJECT IDENTIFIER – OID** в системах управления для обозначения управляемых объектов в MIB.

В этом случае OID кодируется последовательностью целых чисел, указывающих путь к ИЭ в MIB.

Правила кодирования идентификаторов объектов будут рассмотрены в разделе 2.3.3 – структура управляющей информации.

2.4.1 Структура управляющей информации

Для упорядоченной классификации управляющей информации организациями ISO и ITU-T была предложены структура управляющей информации в виде иерархической древовидной системы, растущей от корня дерева (root) вниз.

В рекомендациях ITU-T X.208…X.209 стандартизована вершина глобального дерева (дерево наследования), представленная на рис. 13. Все информационные элементы (объекты управления) в ветвях дерева имеют свой номер, согласно регистрации этого объекта в рамках соответствующей организации.

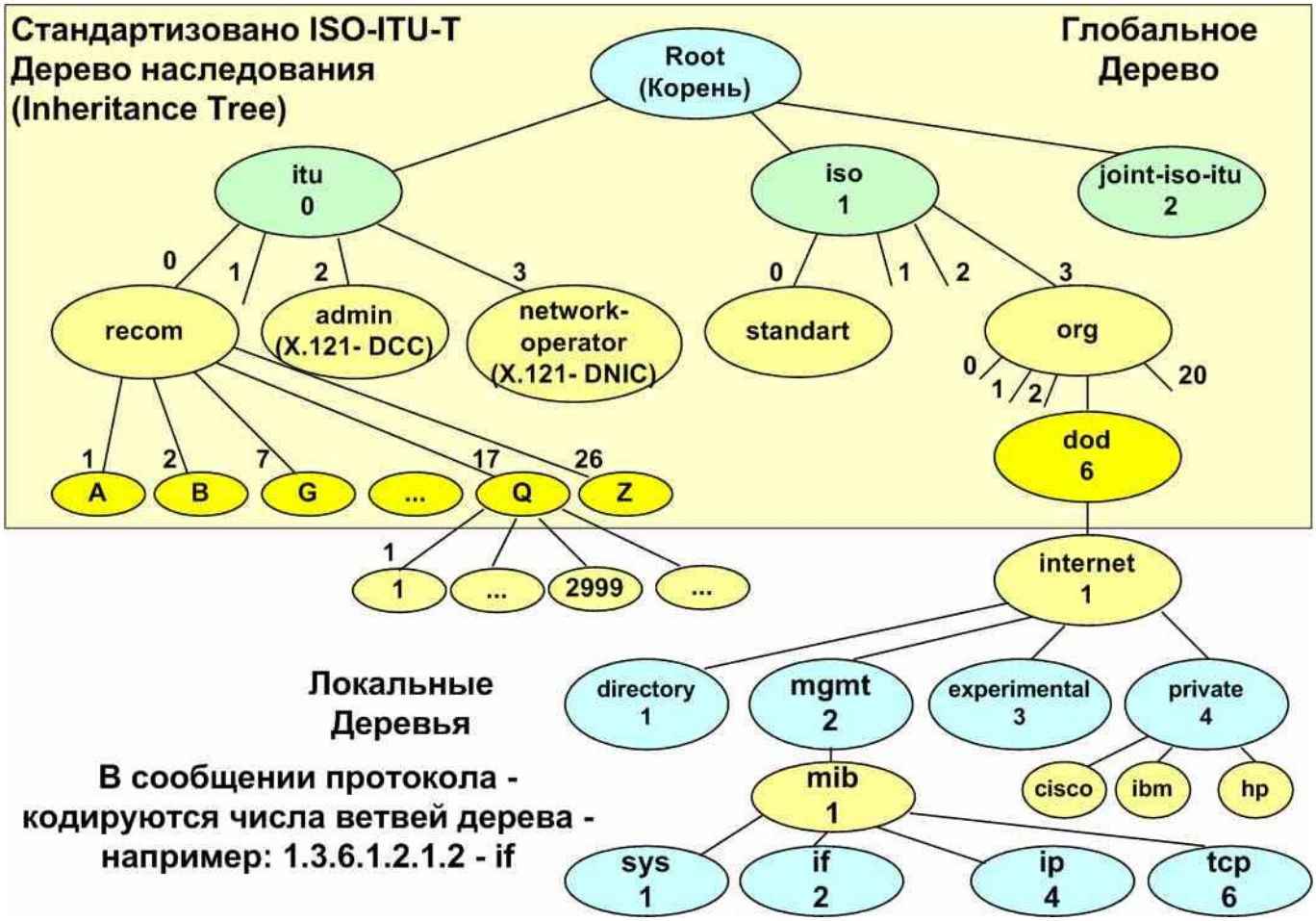


Рисунок 13 – Дерево объектов MIT

Система организации объектов MIT аналогична системе доменных имен, являющимися объектами для другой известной базы данных – таблице доменных имен – DNS.

При кодировании ИЭ, относящихся к вершине дерева MIT есть исключение.

Например, чтобы указать **путь к объекту – MIB** (дерево iso.dod.internet.mgmt, см. рис. 13), **вершина MIT** записывается в виде последовательности следующих целых чисел, разделенных точкой (подчеркнута вершина MIT):

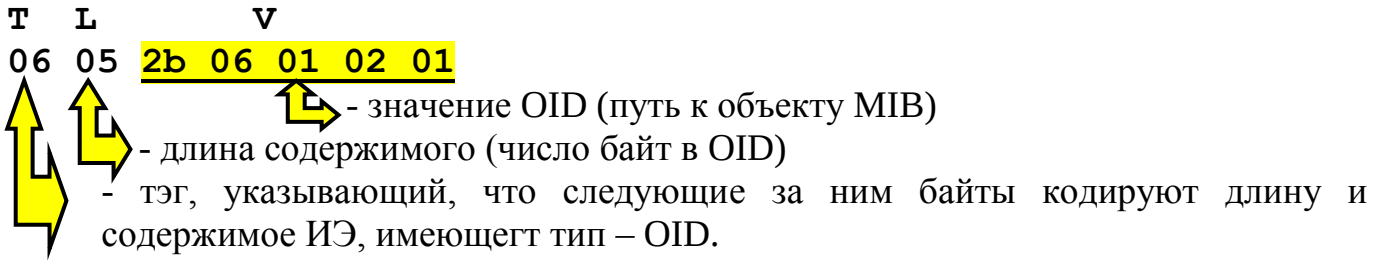
1.3.6.1.2.1 – числовая форма идентификатора объекта в дереве MIT.

Поскольку **заранее известно**, что первое число в вершине дерева (**x**) всегда равно 0 (itu-t), 1 (iso) или 2 (joint-iso-itu), а второе число (**y**) меньше 40, то при передаче **Идентификатора объекта** можно уменьшить количество передаваемой информации, если первые два числа, идентифицирующие вершину дерева MIT, закодировать одним байтом.

Например, для объекта iso.org (x=1.y=3) в вершине дерева сокращенная форма записи выглядит в виде:

40**x** + **y** = 40*1 + 3 =43'Dec или **2B**'Hex.

Остальные числа могут быть больше 256, поэтому **Идентификатор объекта,** находящегося ниже, например, **объект MIB** (iso.org.dod.internet.mgmt.mib, или в числовой нотации 1.3.6.1.2.1) будет передан следующей последовательностью октетов:



В таблице 10 представлены некоторые OID и их значения.

Таблица 10 – Некоторые OID и их значения

Величина OID	Назначение OID
{ 0 0 }	Стандарты ITU-T
{ 1 0 }	Стандарты ISO
{ 1 3 6 }	iso.org.dod – департамент обороны США
{ 1 3 6 1 }	iso.org.dod.internet – объекты сети Интернет
{ 1 2 840 }	iso.member-body. ANSI (US)
{ 2 5 }	Служба каталогов (X.500)
{ 2 5 8 }	Служба каталогов - алгоритмы

2.4.2 Базы данных управляющей информации – MIB

Управляющая система должна точно представлять себе, что и у кого запрашивать. Этого можно достигнуть только в том случае, если управляющей системе – менеджеру во всех деталях известна структура MIB, управляемого сетевого элемента. Напрашивается вывод о том, что должны существовать открытые стандарты на состав и структуру всех MIB.

Однако, такая открытось свойственна только MIB, разработанным в рамках сети Интернет, в частности организацией IETF, и многочисленными поставщиками оборудования для сети Интернет и локальных сетей.

К сожалению, для таких крупных сетевых элементов ТфОП как АТС не существует открытых MIB, что в значительной степени затрудняет посторенние централизованной автоматизированной системы управления ТфОП.

В данной работе мы будем рассматривать детализацию управляемых объектов для сети Интернет.

Существует несколько версий MIB, используемых производителями оборудования и ПО для сетей Интернет и локальных сетей. Приведем некоторые примеры MIB:

1. MIB I (так называемая Internet MIB - RFC 1065, 1066, 1155, 1156, 1157, 1158 и др.) – база данных, определяющая основные имена в дереве MIT, группы Интернет-объектов (ARP, IP. TCP, UDP и т.п.). Обязательна для любого оборудования, обслуживающего сетевые объекты в Интернете. Обеспечивает диагностику ошибок, и конфигурацию различных устройств, оснащенных агентом с MIB-I. Включает в себя около 170 объектов.
2. MIB II (RFC-1213 и др.). Расширяет и детализирует отдельные группы объектов.
3. RMON-1 MIB (RFC 1757). Для управления удаленными объетами вводятся 10 новых групп объектов (см. ниже).
4. RMON-II MIB (RFC 2819). Расширяет количество объектов в приведенных выше группах.

В приведенных MIB постепенно были определены следующие элементы дерева MIT-Internet:

1. Путь к корню глобального дерева iso.org.dod:

```
internet  OBJECT IDENTIFIER ::= { iso(1) org(3) dod(6) 1 }
```

2. Основные имена в локальном дереве Internet:

```
directory     OBJECT IDENTIFIER ::= { internet 1 }
mgmt          OBJECT IDENTIFIER ::= { internet 2 }
experimental  OBJECT IDENTIFIER ::= { internet 3 }
private       OBJECT IDENTIFIER ::= { internet 4 }
```

3. **Для ветви mgmt.mib (2.1.) определены десять групп объектов**, корневые имена (**алиасы**) которых следующие:

```
system      OBJECT IDENTIFIER ::= { mib-2 1 }
interfaces  OBJECT IDENTIFIER ::= { mib-2 2 }
at          OBJECT IDENTIFIER ::= { mib-2 3 }
ip          OBJECT IDENTIFIER ::= { mib-2 4 }
icmp        OBJECT IDENTIFIER ::= { mib-2 5 }
tcp         OBJECT IDENTIFIER ::= { mib-2 6 }
udp         OBJECT IDENTIFIER ::= { mib-2 7 }
egp         OBJECT IDENTIFIER ::= { mib-2 8 }
transmission OBJECT IDENTIFIER ::= { mib-2 10 }
snmp        OBJECT IDENTIFIER ::= { mib-2 11 }
```

Поясним назначение корневых имен поддерева MIB-II:

1. **System** - данная группа MIB II содержит в себе **семь объектов**, каждый из которых служит для хранения информации о системе (версия ОС, время работы и т.д.). Один из объектов этого дерева MIB (а именно - SysUpTime), предстоит узнать в данной работе с помощью протокола SNMP.
2. **Interfaces** - содержит **23 объекта,** необходимых для ведения статистики сетевых интерфейсов агентов (количество интерфейсов, размер MTU, скорость передачи , физические адреса и т.д.) – **именно эту ветвь дерева MIB предстоит в основном исследовать в данной работе с помощью протокола SNMP**.
3. **AT (3 объекта)** - отвечают за трансляцию адресов (Address Translation). Была включена в MIB I. Сейчас почти не используется. Примером использования объектов AT может послужить простая ARP таблица соответствия физических (MAC) адресов сетевых карт IP адресам хостов.
4. **IP (42 объекта)** - данные о проходящих IP пакетах (количество запросов, ответов, отброшенных пакетов).
5. **ICMP (26 объектов)** – информация о контрольных сообщениях (входящие/исходящие сообщения, ошибки и т.д.).
6. **TCP (19 объектов)** - все, что касается одноименного транспортного протокола (алгоритмы, константы, соединения, открытые порты и т.п.).
7. **UDP (6 объектов)** - аналогично, только для UDP протокола (входящие/исходящие датаграммы, порты, ошибки).
8. **EGP (20 объектов)** - данные о трафике Exterior Gateway Protocol (используется маршрутизаторами, объекты хранят информацию о принятых/отосланных/отброшенных кардах).
9. **Transmission** - зарезервирована для специфических MIB.
10. **SNMP (29 объектов)** - статистика по SNMP - входящие/исходящие пакеты, ограничения пакетов по размеру, ошибки, данные об обработанных запросах и многое другое.

Каждый из этих объектов представлен в MIB в виде дерева, растущего вниз. Каждый элемент этого дерева (объект управления) однозначно идентифицируется в этом дереве в форме символьной (**iso.org.dod**….) или цифро-точечной (**1.3.6**…).

Например:

- к адресу администратора мы можем обратиться посредством такого пути: system.syscontact.0,
- ко времени работы системы system.sysUpTime.0,
- к описанию системы (версия, ядро и другая информация об ОС): system.sysDescr.0

С другой стороны те же данные могут задаваться и в цифро-точечной нотации.

Так system.sysUpTime.0 соответствует значение **1.3.0**, так как system имеет индекс "1" в группах MIB II, а sysUpTime - 3 в иерархии группы system.

Ноль в конце пути говорит о скалярном типе хранимых данных (т.е. в данном случае запрашивается **число**, а **не массив данных**).

В RMON-1 MIB (RFC 1757) объекты управления на удаленном оборудовании разделены на следующие группы объектов:
ethernet statistics, history control, ethernet history, alarm, host, hostTopN, matrix filter, packet capture, event.

В приложении 2 приводится полный список объектов MIB II (RFC 1213) из группы if (interface), необходимый для расшифровки сообщений протокола SNMP в данной работе.

2.4.3 Представление SNMP-сообщений

Типы данных, используемые в сообщениях протокола SNMP

Выше были приведены основные типы данных, используемые для описания объектов (информационных элементов) в нотации ASN.1 (см. п.2.3.1 и табл.5).

Синтаксис протокола SNMP, в основном соответствуя нотации ASN.1, допускает использование в своих сообщениях следующих типов данных:

Простые типы данных (Primitive Types) из класса UNI (00):

- **INTEGER - (тэг 02'Hex),**
- **OCTET STRING - (тэг 04'Hex),**
- **NULL - (тэг 05'Hex).**
- **OBJECT IDENTIFIER - (тэг 06'Hex),**
- **Enumerated INTEGER (кроме 0) - (тэг 0a'Hex)**

Составные типы данных (Constructor Types) из класса UNI (00):

- **SEQUENCE** – упорядоченная последовательность (список) из одного или более ИЭ, относящихся к различным типам данных **- (тэг 30'Hex)**
- **SEQUENCE OF** – одномерный массив (таблица) ИЭ одного типа данных **- (тэг 30'Hex)**

Помимо этого (в дополнение к ASN.1), в SNMP определены следующие типы данных (**Defined Types**), относящиеся к прикладному классу **application-wide (01)**:

- **NetworkAddress** – Этот тип данных представляет выбор (Choice) из сетевых адресов одного или нескольких семейств протоколов (например X.25, IP, CCS-7 и т.д.). На данный момент определен один класс сетевых адресов – это **IpAddress.** Этот ИЭ определен тэгом прикладного класса (app или 01) и представляет IP-адрес длиной 32 бита (т.е. IPv4). Адрес представлен как СТРОКА из 4-х ОКТЕТОВ (**OCTET STRING**). При BER-кодировании адреса используется только примитивная форма. Тэг для ИЭ, характеризующего IP-адрес определен как 01 0 00000'bin или **40'hex.**
- **Counter (счетчик)** – Этот тип данных представляет неотрицательное целое число, которое монотонно увеличивается, пока не достигает максимального значения, после чего обнуляется и начинает увеличиваться снова с ноля. Максимальное значение счетчика определено как 2^{32}-1 (4294967295'dec). При BER-кодировании этого ИЭ

используется только примитивная форма. Тэг для ИЭ, характеризующего **Counter** определен как 01 0 00001’bin или **41’hex.**

- **Gauge (размер, величина)** – Этот тип данных представляет неотрицательное целое число, которое может увеличиться или уменьшиться, но **фиксируется** в максимальном значении. Максимальное значение счетчика определено как 2^{32}-1 (4294967295’dec). При BER-кодировании этого ИЭ используется только примитивная форма. Тэг для ИЭ, характеризующего **Gauge** определен как 01 0 00010’bin или **42’hex**
- **TimeTicks (временная метка)** – Этот тип данных представляет неотрицательное целое число, которое обозначает время в сотых долях секунды начиная с некоторого события. При BER-кодировании этого ИЭ используется только примитивная форма. Тэг для ИЭ, характеризующего **TimeTicks** определен как 01 0 00011’bin или **43’hex**
- **Opaque (Непрозрачный, мутный)** – Этот тип ИЭ поддерживает способность передавать произвольный ASN.1 синтаксис (например, зашифрованные данные). Значение кодируется, используя ASN.1 и BER как **строка октетов (OCTET STRING).** Тэг для ИЭ, характеризующего **Opaque** определен как 01000100’bin или **44’hex.** Для расшифровки типа данных **Opaque** обе взаимодействующие стороны должны поддерживать соответствующий синтаксис.

Приведем вышеописанные типы данных, используемые в протоколе SNMP, а также в соответствующих MIB при описании ИЭ, в форме таблицы:

Таблица 11 – Типы данных ИЭ, используемые в протоколе SNMP и MIB

n/n	Тип данных	Тэг	
		‘hex	‘bin
Простые типы данных (**p**rimitive)			
1	INTEGER	**02**	00 **0** 0 0010
2	OCTET STRING	**04**	00 **0** 0 0100
3	NULL	**05**	00 **0** 0 0101
4	OBJECT IDENTIFIER (**OID**)	**06**	00 **0** 0 0110
5	Enumerated INTEGER	**0a**	00 **0** 0 1010
6	**IpAddress**	**40**	01 **0** 0 0000
7	**Counter**	**41**	01 **0** 0 0001
8	**Gauge**	**42**	01 **0** 0 0010
9	**TimeTicks**	**43**	01 **0** 0 0011
10	**Opaque**	**44**	01 **0** 0 0100
Составные типы данных (**c**onstructor)			
11	**SEQUENCE**	**30**	00 **1** 1 0000
12	**SEQUENCE OF**	**30**	00 **1** 1 0000
	Сообщения протокола		
13	**Get**	**a0**	10 **1** 0 0000
14	**Get-Next**	**a1**	10 **1** 0 0001
15	**Response**	**a2**	10 **1** 0 0010
16	**Set**	**a3**	10 **1** 0 0011
17	**Trap**	**a4**	10 **1** 0 0100

Порядок передачи элементов сообщений протокола SNMP

Правила кодирования (BER – X.209) задают структуру тэга для **<u>каждого ИЭ</u>**. Рассмотрим подробнее эти поля:

SNMP-сообщение				Структура ИЭ, входящих в SNMP-сообщение
ИЭ 1			**T**	**TAG**
ИЭ 2				
Информационный элемент (ИЭ i)			**L**	**Длина (Length)**
			V	**Значение (Value, Content)**
…				
ИЭ n				

Рисунок 2.13 – Структура SNMP-сообщения и входящих в него ИЭ

В SNMP-PDU принят следующий порядок передачи:

Порядок передачи бит и байт в сообщениях SNMP

Номер байта	Номер бита в байте							
	MSB	Биты 7...2						LSB
1	8	7	6	5	4	3	2	1
2	8	7	6	5	4	3	2	1
3	8	7	6	5	4	3	2	1
...	...							

MSB – Наиболее значащий бит
LSB – Наименее значащий бит

Биты в каждом байте передаются начиная с LSB – наименее значащего бита (младшего)
Байты передаются начиная с 1-го и т.д.

Рисунок 2.14 – Порядок передачи бит и байт в сообщениях SNMP

Например, в SNMP-последовательности 30 81 fb 02 01 00 04 06 …, представленной ниже в таблице в двоичном коде, первым будет передавться байт 30, затем 81, затем fb, затем 02 и т.д.

Внутри каждого байта сначала передается младший бит (1-й), в данном случае это 0, затем 2-й (0), затем 3-й (0), затем 4-й (0), затем пятый (1) и т.д.

Основные понятия протокола SNMP в нотации ASN.1

Используя вышеприведенные сведения об языке ASN.1, сообщениях протокола SNMP и типах данных, представим основные сведения о протоколе SNMP в форме нотации ASN.1.

1. **Общий формат** сообщения SNMP в **нотации ASN.1** выглядит следующим образом:

```
RFC1157-SNMP DEFINITIONS ::= BEGIN
    SNMP-Message ::=
        SEQUENCE {
            version
                INTEGER {
                version-1 (0)
            },
            community
                OCTET STRING,
            SNMP-PDUs
                ANY
    }
```

2. **Область данных** протокола SNMP может содержать **пять** различных типов протокольных блоков данных (**PDU**), соответствующих **пяти командам** протокола SNMP:

```
    SNMP-PDUs ::=
        CHOICE {
            get-request
                GetRequest-PDU,
            get-next-request
                GetNextRequest-PDU,
            get-response
                GetResponse-PDU,
            set-request
                SetRequest-PDU,
            trap
                Trap-PDU,
    }
```

3. Для каждого типа PDU имеется определение его **формата**. Например, формат блока GetRequest-PDU описан следующим образом:

```
    GetRequest-PDU ::=
        IMPLICIT SEQUENCE {
            request-id
                RequestID,
            error-status
                ErrorStatus,
            error-index
                Errorlndex,
            variable-bindings
                VarBindList
                }
```

2.3.5.4. Определение элементов в общей конструкции PDU-SNMP

Далее для каждого формата следуют определения элементов PDU. Например, для запросов и ответов определены следующие элементы и соответствующие типы данных:

```
    -- request/response information

                    RequestID ::=
```

```
                INTEGER
        ErrorStatus ::=
                INTEGER {
                    noError(0),
                    tooBig(1),
                    noSuchName(2),
                    badValue(3),
                    readOnly(4)
                    genErr(5)
                }
        ErrorIndex ::=
                INTEGER
    -- variable bindings
        VarBind ::=
                SEQUENCE {
                    name
                        ObjectName,
                    value
                        ObjectSyntax
                }
        VarBindList ::=
                SEQUENCE OF
                    VarBind
1.  The GetRequest-PDU
        GetRequest-PDU ::=
            [0]
                IMPLICIT SEQUENCE {
                    request-id
                        RequestID,
                    error-status        -- always 0
                        ErrorStatus,
                    error-index         -- always 0
                        ErrorIndex,
                    variable-bindings
                        VarBindList
                }
2.  The GetNextRequest-PDU
        GetNextRequest-PDU ::=
            [1]
                IMPLICIT SEQUENCE {
                    request-id
                        RequestID,
                    error-status        -- always 0
                        ErrorStatus,
                    error-index         -- always 0
                        ErrorIndex,
                    variable-bindings
                        VarBindList
                }
3.  The GetResponse-PDU
        GetResponse-PDU ::=
            [2]
                IMPLICIT SEQUENCE {
```

```
                    request-id
                        RequestID,
                    error-status
                        ErrorStatus,
                    error-index
                        ErrorIndex,
                    variable-bindings
                        VarBindList
                }
```

4. The SetRequest-PDU

```
        SetRequest-PDU ::=
            [3]
                IMPLICIT SEQUENCE {
                    request-id
                        RequestID,
                    error-status        -- always 0
                        ErrorStatus,
                    error-index         -- always 0
                        ErrorIndex,
                    variable-bindings
                        VarBindList
                }
```

5. The Trap-PDU

```
        Trap-PDU ::=
              [4]
            IMPLICIT SEQUENCE {
                enterprise          -- type of object generating
                                    -- trap, see sysObjectID in [5]
                    OBJECT IDENTIFIER,
                agent-addr          -- address of object generating
                    NetworkAddress, -- trap
                generic-trap        -- generic trap type
                    INTEGER {
                        coldStart(0),
                        warmStart(1),
                        linkDown(2),
                        linkUp(3),
                        authenticationFailure(4),
                        egpNeighborLoss(5),
                        enterpriseSpecific(6)
                    },
                specific-trap       -- specific code, present even
                    INTEGER,        -- if generic-trap is not
                                    -- enterpriseSpecific
                time-stamp          -- time elapsed between the last
                  TimeTicks,        -- (re)initialization of the network
                                    -- entity and the generation of the
                                       trap
                variable-bindings   -- "interesting" information
                     VarBindList
            }
```

2.5 Методические указания к расшифровке протокола SNMP

Варианты заданий

Варианты заданий к курсовой работе приводятся **в папке Variant**.

Здесь приведем один из вариантов задания и пример его расшифровки.

Вариант задания № n

Заданы следующие сообщения:

1. Сообщение №1

```
0000: 00 00 1d 90 58 20 00 20   af e8 e2 8e 08 00 45 00
0010: 01 1a 0b 25 00 00 40 11   00 09 d4 a4 00 66 d4 a4
0020: c4 f6 c0 7c 00 a1 01 06   4a 51 30 81 fb 02 01 00
0030: 04 06 76 6d 31 35 2d 31   a0 81 ed 02 04 35 97 ac
0040: 55 02 01 00 02 01 00 30   81 de 30 0c 06 08 2b 06
0050: 01 02 01 01 03 00 05 00   30 0e 06 0a 2b 06 01 02
0060: 01 02 02 01 05 01 05 00   30 0e 06 0a 2b 06 01 02
0070: 01 02 02 01 08 01 05 00   30 0e 06 0a 2b 06 01 02
0080: 01 02 02 01 09 01 05 00   30 0e 06 0a 2b 06 01 02
0090: 01 02 02 01 0a 01 05 00   30 0e 06 0a 2b 06 01 02
00a0: 01 02 02 01 0b 01 05 00   30 0e 06 0a 2b 06 01 02
00b0: 01 02 02 01 0c 01 05 00   30 0e 06 0a 2b 06 01 02
00c0: 01 02 02 01 0d 01 05 00   30 0e 06 0a 2b 06 01 02
00d0: 01 02 02 01 0e 01 05 00   30 0e 06 0a 2b 06 01 02
00e0: 01 02 02 01 10 01 05 00   30 0e 06 0a 2b 06 01 02
00f0: 01 02 02 01 11 01 05 00   30 0e 06 0a 2b 06 01 02
0100: 01 02 02 01 12 01 05 00   30 0e 06 0a 2b 06 01 02
0110: 01 02 02 01 13 01 05 00   30 0e 06 0a 2b 06 01 02
0120: 01 02 02 01 14 01 05 00
```

2. Сообщение №2

```
0000: 00 20 af e8 e2 8e 00 00   1d 7c 63 f1 08 00 45 00
0010: 01 37 9c bf 00 00 3e 11   70 56 d4 a4 c4 f1 d4 a4
0020: 00 66 00 a1 c0 7a 01 23   7b 84 30 82 01 17 02 01
0030: 00 04 05 65 78 70 2d 31   a2 82 01 09 02 04 35 97
0040: ac 59 02 01 00 02 01 00   30 81 fa 30 0f 06 08 2b
0050: 06 01 02 01 01 03 00 43   03 73 d4 70 30 11 06 0a
0060: 2b 06 01 02 01 02 02 01   05 03 42 03 00 fa 00 30
0070: 0f 06 0a 2b 06 01 02 01   02 02 01 08 03 02 01 01
0080: 30 0f 06 0a 2b 06 01 02   01 02 02 01 09 03 43 01
0090: 00 30 12 06 0a 2b 06 01   02 01 02 02 01 0a 03 41
00a0: 04 04 12 5a 5d 30 11 06   0a 2b 06 01 02 01 02 02
00b0: 01 0b 03 41 03 08 6f da   30 0f 06 0a 2b 06 01 02
00c0: 01 02 02 01 0c 03 41 01   07 30 0f 06 0a 2b 06 01
00d0: 02 01 02 02 01 0d 03 41   01 00 30 0f 06 0a 2b 06
00e0: 01 02 01 02 02 01 0e 03   41 01 00 30 12 06 0a 2b
00f0: 06 01 02 01 02 02 01 10   03 41 04 13 a1 03 ca 30
0100: 11 06 0a 2b 06 01 02 01   02 02 01 11 03 41 03 08
0110: 0d 32 30 0f 06 0a 2b 06   01 02 01 02 02 01 12 03
0120: 41 01 00 30 0f 06 0a 2b   06 01 02 01 02 02 01 13
0130: 03 41 01 00 30 0f 06 0a   2b 06 01 02 01 02 02 01
0140: 14 03 41 01 00
```

В сообщении №1 администратором сети **запрашивается** информация об управляемом объекте:

В сообщении №2 в ответном сообщении Response от агента **доставляется** информация об управляемом объекте:

1. Задание:

Расшифровать приведенные в hex'кодах сообщения управляющего протокола, в соответствии с поставленными ниже в пп. 1…18 вопросами.

Ответы оформить в соответствии с прилагаемыми ниже требованиями.

Для расшифровки сообщений используйте сведения в прилагаемых файлах – rfc1213, rfc1700, ETHERNET VENDOR ADDRESS.doc, ETHER TYPES.doc, а также сведения, полученные на лекциях и практических занятиях.

2. Определить из приведенных сообщений:

1. Фирму-поставщика оборудования сетевых интерфейсов
2. MAC-адреса источника и назначения
3. Тип протокола, обслуживаемого данным Ethernet-кадром
4. Версию протокола сетевого уровня
5. Приоритет сетевого уровня для данной дейтаграммы
6. Длину пакета сетевого уровня (в байтах)
7. Время жизни данной дейтаграммы
8. Протокол транспортного уровня (Dec'код и название)
9. Сетевой адрес отправителя
10. Сетевой адрес назначения
11. Транспортный порт отправителя
12. Транспортный порт получателя
13. Тип и версию протокола прикладного уровня
14. Длину дейтаграммы транспортного уровня (в байтах)
15. Тип и класс тэга протокола прикладного уровня
16. Длину сообщения протокола прикладного уровня
17. Длину и содержимое поля Community
18. Тип PDU и его длину (в байтах)
    - 18.1. Для PDU типа **Get-Request**
      - 18.1.1. Значение идентификатора запроса - **RequestID**
      - 18.1.2. Значения полей **ErrorStatus и Errorlndex**
      - 18.1.3. Длину поля, содержащего набор запрашиваемых характеристик
      - 18.1.4. Перечень запрашиваемых характеристик (атрибутов) управляемого объекта*
    - 18.2. Для PDU типа **GetResponse**
      - 18.2.1. Значение идентификатора запроса – **RequestID**
      - 18.2.2. Значения полей **ErrorStatus и Errorlndex**
      - 18.2.3. Длину поля, содержащего набор характеристик управляемого объекта
      - 18.2.4. Перечень характеристик (атрибутов) управляемого объекта*
      - 18.2.5. Значения характеристик (атрибутов) управляемого объекта*

Примечание:

1. Ответы на эти 18 вопросов оформить в виде таблиц, как показано в п.3.3 – Правила оформления контрольной работы.
2. Ответы на вопросы в пунктах, отмеченных (*), привести в виде отдельной таблицы (см.п 3.3)

Итак, в сообщении №1 приведена следующая 16-ричная трассировка:

```
0000: 00 00 1d 90 58 20 00 20   af e8 e2 8e 08 00 45 00
0010: 01 1a 0b 25 00 00 40 11   00 09 d4 a4 00 66 d4 a4
0020: c4 f6 c0 7c 00 a1 01 06   4a 51 30 81 fb 02 01 00
0030: 04 06 76 6d 31 35 2d 31   a0 81 ed 02 04 35 97 ac
0040: 55 02 01 00 02 01 00 30   81 de 30 0c 06 08 2b 06
0050: 01 02 01 01 03 00 05 00   30 0e 06 0a 2b 06 01 02
0060: 01 02 02 01 05 01 05 00   30 0e 06 0a 2b 06 01 02
0070: 01 02 02 01 08 01 05 00   30 0e 06 0a 2b 06 01 02
0080: 01 02 02 01 09 01 05 00   30 0e 06 0a 2b 06 01 02
0090: 01 02 02 01 0a 01 05 00   30 0e 06 0a 2b 06 01 02
00a0: 01 02 02 01 0b 01 05 00   30 0e 06 0a 2b 06 01 02
00b0: 01 02 02 01 0c 01 05 00   30 0e 06 0a 2b 06 01 02
00c0: 01 02 02 01 0d 01 05 00   30 0e 06 0a 2b 06 01 02
00d0: 01 02 02 01 0e 01 05 00   30 0e 06 0a 2b 06 01 02
00e0: 01 02 02 01 10 01 05 00   30 0e 06 0a 2b 06 01 02
00f0: 01 02 02 01 11 01 05 00   30 0e 06 0a 2b 06 01 02
0100: 01 02 02 01 12 01 05 00   30 0e 06 0a 2b 06 01 02
0110: 01 02 02 01 13 01 05 00   30 0e 06 0a 2b 06 01 02
0120: 01 02 02 01 14 01 05 00
```

Отобразим этот пример трассировки, **выделив разными цветами** заголовки стека протоколов SNMP/UDP/IP/Ethernet, обеспечивающих передачу сообщений SNMP по сети IP.

Пример трассировки сообщений протокола **SNMP**, включая заголовки транспортных протоколов **UDP/IP/Ethernet** (в Hex' коде):

```
0000: 00 00 1d 90 58 20 00 20   af e8 e2 8e 08 00 45 00
0010: 01 1a 0b 25 00 00 40 11   00 09 d4 a4 00 66 d4 a4
0020: c4 f6 c0 7c 00 a1 01 06   4a 51 30 81 fb 02 01 00
0030: 04 06 76 6d 31 35 2d 31   a0 81 ed 02 04 35 97 ac
0040: 55 02 01 00 02 01 00 30   81 de 30 0c 06 08 2b 06
0050: 01 02 01 01 03 00 05 00   30 0e 06 0a 2b 06 01 02
0060: 01 02 02 01 05 01 05 00   30 0e 06 0a 2b 06 01 02
0070: 01 02 02 01 08 01 05 00   30 0e 06 0a 2b 06 01 02
0080: 01 02 02 01 09 01 05 00   30 0e 06 0a 2b 06 01 02
0090: 01 02 02 01 0a 01 05 00   30 0e 06 0a 2b 06 01 02
00a0: 01 02 02 01 0b 01 05 00   30 0e 06 0a 2b 06 01 02
00b0: 01 02 02 01 0c 01 05 00   30 0e 06 0a 2b 06 01 02
00c0: 01 02 02 01 0d 01 05 00   30 0e 06 0a 2b 06 01 02
00d0: 01 02 02 01 0e 01 05 00   30 0e 06 0a 2b 06 01 02
00e0: 01 02 02 01 10 01 05 00   30 0e 06 0a 2b 06 01 02
00f0: 01 02 02 01 11 01 05 00   30 0e 06 0a 2b 06 01 02
0100: 01 02 02 01 12 01 05 00   30 0e 06 0a 2b 06 01 02
0110: 01 02 02 01 13 01 05 00   30 0e 06 0a 2b 06 01 02
0120: 01 02 02 01 14 01 05 00
```

Рисунок 2.15 – Трассировка SNMP-сообщения

Общая длина приведенного сообщения – 296 байт и складывается из следующих частей (поля протокола стека **SNMP**/**UDP**/**IP**/**Ethernet** выделены цветом):

296 байт = 14 Ethernet + 20 IP + 8 UDP + 254 SNMP

В данной трассировке администратором сети запрашивается следующая информация об управляемом объекте:

```
system.sysUpTime.0
interfaces.ifTable.ifEntry.ifSpeed.1
interfaces.ifTable.ifEntry.ifOperStatus.1
interfaces.ifTable.ifEntry.ifLastChange.1
interfaces.ifTable.ifEntry.ifInOctets.1
interfaces.ifTable.ifEntry.ifInUcastPkts.1
interfaces.ifTable.ifEntry.ifInNUcastPkts.1
interfaces.ifTable.ifEntry.ifInDiscards.1
interfaces.ifTable.ifEntry.ifInErrors.1
interfaces.ifTable.ifEntry.ifOutOctets.1
interfaces.ifTable.ifEntry.ifOutUcastPkts.1
interfaces.ifTable.ifEntry.ifOutNUcastPkts.1
interfaces.ifTable.ifEntry.ifOutDiscards.1
interfaces.ifTable.ifEntry.ifOutErrors.1
```

В данном пособии детально рассматривается расшифровка всех полей стека протоколов **SNMP/UDP/IP/Ethernet**.

Начнем расшифровку данного примера трассировки с протоколов нижних уровней.

Поля протокола **Ethernet**: `00 00 1d 90 58 20 00 20 af e8 e2 8e 08 00`

1. Фрагмент трассировки заголовка Ethernet в Hex'-коде

00	00	1d	90	58	20	00	20	af	e8	e2	8e	08	00

2. Формат заголовка протокола Ethernet (всего - 14 байт)

MAC-DA (Адрес сетевой платы назначения) 6 байт	MAC-SA (Адрес сетевой платы источника) 6 байт	Length/ Type (Protocol) 2 байта

3. Кодировка полей протокола Ethernet

Vendor 3 байта	Serial Number 3 байта	Vendor 3 байта	Serial Number 3 байта	dod IP 2 байта

Первые 3 байта **MAC-адресов** отведены для кода фирмы (вендора), выпускающей данное оборудование. Некоторые коды фирм приведены в RFC-1700 (см. RFC-1700, раздел – **ethernet vendor address components**, а также приложение 4 к данному пособию). Согласно этому разделу расшифруем коды вендоров:

`00 00 1d` – Сетевой интерфейс фирмы **Cabletron**
`00 20 af` – Сетевой интерфейс фирмы **3COM**

Последние 3 байта **MAC-адресов** отведены для серийного номера конкретной сетевой платы, который может быть назначен динамически, запрограммирован вендором или устанавливаться администратором сети.

Последние два байта заголовка протокола Ethernet, кодируют либо длину Ethernet-кадра (для версий IEEE 802.3), либо тип обслуживаемого протокола вышележащего уровня (для версий Ethernet II).

В таблице 2.1 приведены некоторые коды этого поля. Полный перечень см. в RFC-1700 (в разделе – **ether types,** а также в приложении 5).

Таблица 2.1 - Некоторые коды протоколов в Ethernet II (см. RFC-1700)

Десятичный код	Hex	Описание
0 … 1500	00 00 … 05 DC	Поле длины IEEE 802.3
2048	08 00	Internet протокол (IPv4)
2049	08 01	X.75 Internet
2053	08 05	X.25 уровень 3
2054	08 06	Протокол трансляции адреса (ARP)
32821	80 35	Реверсивный протокол ARP (RARP)
33079	81 37-81 38	NetWare IPX/SPX
33100	81 4C	SNMP over Ethernet (см. RFC-1089)

Расшифруем эти два байта в нашем случае:

08 00 – Эта кодировка означает, что данный Ethernet-кадр перевозит в поле данных IP-датаграмму (данные **Internet-протокола версии 4 (IPv4)**.

Поля протокола IP (заголовок IP-датаграммы)

Формат заголовка протокола IPv4 (5 слов по 32 бита):

<table>
<tr><td colspan="4">0 1 2 3</td><td colspan="4">4 5 6 7</td><td colspan="8">8 9 10 11 12 13 14 15</td><td colspan="16">16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31</td></tr>
<tr><td colspan="4">Версия</td><td colspan="4">Длина IP-заголовка (HLength)</td><td colspan="8">Тип сервиса
ToS
0 1 2 3 4 5 6 7
Приоритет | D | T | R | C | Не использ.</td><td colspan="16">Длина IP-пакета (дейтаграммы), включая заголовки IP и UDP</td></tr>
<tr><td colspan="16">Идентификатор фрагмента</td><td colspan="3">Флаги</td><td colspan="13">Указатель фрагмента</td></tr>
<tr><td colspan="8">Время жизни (TTL)</td><td colspan="8">Протокол, которому предоставлена услуга</td><td colspan="16">Контрольная сумма заголовка</td></tr>
<tr><td colspan="32">IP-адрес отправителя – Source (откуда)</td></tr>
<tr><td colspan="32">IP-адрес получателя – Destination (куда)</td></tr>
</table>

Приведем расшифровку заголовка IP из трассировки сообщения №1:

```
                                                          45 00
0010: 01 1a 0b 25 00 00 40 11   00 09 d4 a4 00 66 d4 a4
0020: c4 f6
```

Ниже в таблице приведена расшифровка заголовка IP-датаграммы:

0 1 2 3	4 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Версия	Длина IP-заголовка (HLength)	**Тип сервиса** ToS 0 1 2 3 4 5 6 7 Приоритет D T R C Не использ.	**Длина IP-пакета** (дейтаграммы), включая заголовки IP и UDP
4	5	00	01 1a‘hex=282’Dec (байт)
		Prio D T R C x	
		0 0 0 0 0 0 0 0	

Идентификатор фрагмента	Флаги	Указатель фрагмента
0b 25		00 00
	0 0 0	

Время жизни (TTL)	**Протокол**, которому предоставлена услуга	**Контрольная сумма заголовка**
40’hex (64’Dec)	11’hex (17’Dec - UDP)	00 09

IP-адрес отправителя – Source (откуда)			
d4’hex 212’Dec	a4’hex 164’Dec	00’hex 00’Dec	66’hex 102’Dec
IP-адрес получателя – Destination (куда)			
d4’hex 212’Dec	a4’hex 164’Dec	c4’hex 196’Dec	f6’hex 246’Dec

Из расшифровки видно, что IP-пакет длиной 282 байта, перевозящий данное SNMP-сообщение, направляется от устройства с адресом IP – 212.164.00.102 к устройству с адресом IP – 212.164.196.246, при этом время жизни IP-пакета в сети ограничено значением TTL=64, что допускает 64 транзитных пункта.

Также видно, что приоритет данного пакета самый низкий (0), и для обслуживания SNMP-сообщения используется ненадежный протокол UDP (код – 11’hex или 17’dec).

Поле **контрольная сумма** заголовка вычисляется с использованием операций сложения 16-разрядных слов **заголовка** по модулю 1.

Обратите внимание, здесь осуществляется контрольное суммирование слов **заголовка, а не всей дейтаграммы**.

Поля протокола UDP (Заголовок UDP-датаграммы)

Фрагмент (заголовок) UDP-датаграммы: `c0 7c 00 a1 01 06 4a 51`

Формат заголовка протокола UDP:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Порт отправителя (от кого)																**Порт назначения (кому)**															
Длина UDP-пакета																**Контрольная сумма заголовка**															

Полный перечень номеров UDP-портов приведен в RFC-1700.

Номера портов от 0 до 255 стандартизованы и закреплены за наиболее известными сервисными протоколами.

Номер порта 161 закреплен за сообщениями протокола SNMP. Для SNMP-сообщений типа Trap, выделен порт с номером 162. Использовать эти «очень известные порты» в прикладных задачах не рекомендуется.

В интервале 255-1023 многие номера портов также заняты, поэтому прежде чем использовать какой-то порт в своей программе, следует заглянуть в RFC-1700.

Номера портов с 4096 по 65536, назначаются динамически по согласованию сторон, участвующих в обмене информацией.

Длина сообщения (**UDP-пакета**) равна числу байт в UDP-датаграмме, включая заголовок UDP.

Поле UDP контрольная сумма содержит код, полученный в результате контрольного суммирования **UDP-заголовка и поля «данные»**.

Ниже в таблице приведена расшифровка заголовка UDP-датаграммы:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
1-ое 32-х разрядное слово UDP-заголовка																															
Порт отправителя (от кого)																**Порт назначения (кому)**															
c0 7c (49351'Dec)																00 a1 (161'Dec - SNMP)															
2-ое 32-х разрядное слово UDP-заголовка																															
Длина UDP-пакета																**Контрольная сумма заголовка**															
01 06 (262'Dec байт)																4a 51															

Из данной расшифровки видно, что UDP-дейтаграмма, обслуживающая SNMP-сообщение, имеет общую длину 262 байта и предназначена для приложения с портом 161. Со стороны источника используется динвмически назначенный порт с номером 49351.

Итак, расшифрованную информацию из заголовков Ethernet/IP/UDP сведем в таблицу по прилагаемой в задании форме (т.е. заполним в твблице ответы на первые 14 вопросов):

Определить из приведенных сообщений

1. Фирму-поставщика оборудования сетевых интерфейсов
2. MAC-адреса источника и назначения
3. Тип протокола, обслуживаемого данным Ethernet-кадром
4. Версию протокола сетевого уровня
5. Приоритет сетевого уровня для данной дейтаграммы
6. Длину пакета сетевого уровня (в байтах)
7. Время жизни данной дейтаграммы
8. Протокол транспортного уровня (Dec'код и название)
9. Сетевой адрес отправителя
10. Сетевой адрес назначения
11. Транспортный порт отправителя
12. Транспортный порт получателя

13. Тип и версию протокола прикладного уровня
14. Длину дейтаграммы транспортного уровня (в байтах)

№ вопроса	Для PDU типа **Get-Request**		Для PDU типа **Get-Response**	
	Hex' значение	Dec' или текстовое значение	Hex' значение	Dec' или текстовое значение
1	00 00 1d 00 20 af	– Сетевой интерфейс фирмы **Cabletron** – Сетевой интерфейс фирмы **3COM**		
2	00 00 1d 90 58 20 00 20 af e8 e2 8e	MAC-адрес назначения MAC-адрес источника		
3	08 00	протокол IPv4		
4	4	4-я версия		
5	00	000 – низкий приоритет		
6	01 1a	282 байта		
7	40	TTL=64 транзита		
8	11	17 – UDP протокол		
9	d4 a4 00 66	212.164.00.102		
10	d4 a4 c4 f6	212.164.196.246		
11	c0 7c	49351 - DP		
12	00 a1	161 - SNMP		
13	00 a1	161 - SNMP		
14	01 06	262 байта		
…				

Аналогично определяем и заполняем таблицу для второго сообщения!

Поля протокола SNMP

Наконец, мы добрались до расшифровки самого SNMP-сообщения. Чтобы разобрать отдельные его составляющие, приведем все поля, относящиеся к протоколу SNMP отдельно.

Итак, фрагмент SNMP-сообщения, вложенного в информационную часть протоколов UDP/IP/Ethernet выглядит так:

```
                                                  30 81 fb 02 01 00
0030: 04 06 76 6d 31 35 2d 31   a0 81 ed 02 04 35 97 ac
0040: 55 02 01 00 02 01 00 30   81 de 30 0c 06 08 2b 06
0050: 01 02 01 01 03 00 05 00   30 0e 06 0a 2b 06 01 02
0060: 01 02 02 01 05 01 05 00   30 0e 06 0a 2b 06 01 02
0070: 01 02 02 01 08 01 05 00   30 0e 06 0a 2b 06 01 02
0080: 01 02 02 01 09 01 05 00   30 0e 06 0a 2b 06 01 02
0090: 01 02 02 01 0a 01 05 00   30 0e 06 0a 2b 06 01 02
00a0: 01 02 02 01 0b 01 05 00   30 0e 06 0a 2b 06 01 02
00b0: 01 02 02 01 0c 01 05 00   30 0e 06 0a 2b 06 01 02
00c0: 01 02 02 01 0d 01 05 00   30 0e 06 0a 2b 06 01 02
00d0: 01 02 02 01 0e 01 05 00   30 0e 06 0a 2b 06 01 02
00e0: 01 02 02 01 10 01 05 00   30 0e 06 0a 2b 06 01 02
00f0: 01 02 02 01 11 01 05 00   30 0e 06 0a 2b 06 01 02
0100: 01 02 02 01 12 01 05 00   30 0e 06 0a 2b 06 01 02
0110: 01 02 02 01 13 01 05 00   30 0e 06 0a 2b 06 01 02
0120: 01 02 02 01 14 01 05 00
```

Для упрощения расшифровки данного сообщения, разобъем это сообщение на отдельные составляющие в соответствии с форматом, приведенным в разделе 2.2 и на рис. 7, 8 и 9, а также в соответствии с правилами кодирования SNMP-сообщений:

```
30 81 fb
04 06 76 6d 31 35 2d 31
a0 81 ed
02 04 35 97 ac 55
02 01 00
02 01 00
30 81 de
30 0c 06 08 2b 06 01 02 01 01 03 00 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 05 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 08 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 09 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 0a 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 0b 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 0c 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 0d 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 0e 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 10 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 11 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 12 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 13 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 14 01 05 00
```

Учитывая рекомендации, приведенные в разделе 2.3, применим к данному формату конструкцию T-L-V, чтобы расшифровать отдельные ИЭ сообщения SNMP:

Конструкция – T-L-V (Tag-Length-Value)

```
30 81 fb
T    L    (..... V .....
02 01 00
T  L  V
04 06 76 6d 31 35 2d 31
T  L        V

a0 81 ed
T    L    (..... V .....

02 04 35 97 ac 55
T  L       V

02 01 00
T  L  V

02 01 00
T  L  V

30 81 de
T    L    (..... V .....

30 0c 06 08 2b 06 01 02 01 01 03 00 05 00
T  L (T  L (      V=OID                  ))
```

Остальные части кодируются аналогично:

```
30 0e 06 0a 2b 06 01 02 01 02 02 01 05 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 08 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 09 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 0a 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 0b 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 0c 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 0d 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 0e 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 10 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 11 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 12 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 13 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 14 01 05 00
```

Учитываем, что ИЭ, кодируемые тэгом=30 или a0, содержащим в 6-м бите единицу, имеют **составной тип**, следовательно, внутри этих элементов содержатся другие ИЭ по принципу матрешки.

Расшифруем теперь отдельные части SNMP-сообщения:

```
30 81 fb
T  L  (..... V .....
```

– Заголовок протокола SNMP (флаг), содержащий тэг (30 - Sequence) и длину содержимого (81 fb – длинный формат, обозначающий, что в поле «Длина» содержится 1 байт, а его значение – fb'hex или 251 байт)

```
02 01 00
T  L  V
```

– Версия протокола SNMP (Тэг=02, что означает целое число в поле «содержимое», длина этого содержимого равна 1 байту, а 00 в поле содержимое – означает, что используется версия SNMPv1)

```
04 06 76 6d 31 35 2d 31
T  L        V
```

– поле "Community" длиной 6 байт, в котором содержится строка октетов (тэг=04), кодирующих содержимое в формате IA5.

В данном случае поля **76 6d 31 35 2d 31**'hex означают, что пароль доступа к полю "Community" – **vm15-1**

```
a0 81 ed
T  L  (..... V .....
```

– имя PDU-SNMP. В данном случае – это Get-request (тэг=a0), а длина содержимого в этом PDU составляет ed'hex или 237'dec байт

Далее, учитывая, что имя данного PDU – Get, используем его формат (см. рис. 7 и 8 в разделе 2.2, а также форму записи на языке ASN.1 в разделе 2.3.5.3 для заголовка Get-PDU):

<table>
<tr><th colspan="9">Заголовок SNMP</th><th colspan="9">Заголовок PDU
(для Get, GetNext, Set, Response)</th><th colspan="5">Переменные
(Идентификаторы объектов в MIB, значения параметров)</th></tr>
<tr><td colspan="3">Версия</td><td colspan="3">Пароль (Community)</td><td colspan="3">Тип PDU</td><td colspan="3">Идентификатор PDU</td><td colspan="3">Статус ошибки</td><td colspan="3">Индекс ошибки</td><td>Имя (Tag)</td><td>Длина (L)</td><td colspan="3">Значение (Value)</td></tr>
<tr><td colspan="3">Vers – 1</td><td colspan="3">По умолчанию: public</td><td colspan="3">См. табл.1</td><td colspan="3">Целое значение от 0 до 2^{32}-1</td><td colspan="3">См. табл.2</td><td colspan="3">См. ниже</td><td>***</td><td>****</td><td>Имя (Tag)</td><td>Длина (L)</td><td>…</td></tr>
<tr><td>T</td><td>L</td><td>V</td><td>T</td><td>L</td><td>V</td><td>T</td><td>L</td><td>V</td><td>T</td><td>L</td><td>V</td><td>T</td><td>L</td><td>V</td><td>T</td><td>L</td><td>V</td><td>T</td><td>L</td><td>T</td><td>L</td><td>V</td></tr>
</table>

Или в форме ASN.1:

```
The GetRequest-PDU
                GetRequest-PDU ::=
                    [0]
                        IMPLICIT SEQUENCE {
                            request-id
                                RequestID,
                            error-status          -- always 0
                                ErrorStatus,
                            error-index           -- always 0
                                ErrorIndex,
                            variable-bindings
                                VarBindList
                        }
```

```
02 04 35 97 ac 55
T  L      V
```

– идентификатор данного запроса (**request-id**). Используется для того, чтобы связывать запросы и ответы на них в пары. В данном случае длина этого идентификатора равна 4 байтам, а так как поле содержимого кодируется целым числом (тэг=02’hex, что означает INTEGER), то значение идентификатора будет 35 97 ac 55’hex или 899132501’dec

```
02 01 00
T  L  V
```

– статус ошибки (**error-status**). Как указано выше, для запросов это значение всегда=0 (always 0)

```
02 01 00
T  L  V
```

– индекс ошибки (**error-index**). Также как и для статуса, как указано выше, в запросах это значение всегда=0 (always 0)

```
30 81 de
T    L   (..... V .....
```

– Тэг=30 (Sequence), означает, что далее идет составной тип данных (последовательность) длиной de’hex или 222’dec байт. В данном случае, в соответствии с форматом Get-PDU – это последовательность переменных (**variable-bindings**).

Далее следуют переменные, которые собственно и содержат основную управляющую информацию, интересующую менеджера в данном запросе:

```
30 0c 06 08 2b 06 01 02 01 01 03 00 05 00
T  L ( T  L (      V = OID               ))
```

Как видно, что первая переменная представляет собой также последовательность (тэг=30, следовательно, тип ИЭ - составной).

Выделим элементы этой последовательности:

```
30 0c
T  L  (... V ...
```

– последовательность переменных общей длиной 0c’hex или 12 байт

```
06 08 2b 06 01 02 01 01 03 00 05 00
(T  L (      V = OID       ), T  L )
```

– или более детально:

```
06 08
(T  L
```

– Тэг=06, следовательно, 1-я переменная в этой последовательности – это идентификатор объекта – OID, длиной 8 байт

```
2b 06 01 02 01 01 03 00
(      V = OID          ),
```

- содержимое первой переменной (OID), которое в цифро-точечной нотации означает 1.3.6.1.2.1.1.3.0, что означает – менеджер запрашивает значение переменной, находящейся в базе данных (MIB) по пути (см. п.2.3.4):

```
iso.org.dod.internet.mgmt.mib.sys
 1 . 3 . 6 .    1   . 2  . 1 . 1
```

(напомним, что 2b=1.3 – вершина дерева iso.org)

По этому пути для объектов группы **system** находится запрашиваемая переменная – **sysUpTime** (OID=3 в иерархии группы объектов system) – время с момента последней перезагрузки объекта.

Итак, в запросе Get, менеджер спрашивает: сколько времени прошло с момента последней перезагрузки объекта, что выглядит так:

```
iso.org.dod.internet.mgmt.mib.sys.sysUpTime.0
 1 . 3 . 6 .    1   . 2  . 1 . 1 .   3     .0
   2b    06     01    02   01  01    03     00
```

Ноль в конце пути говорит о скалярном типе хранимых данных (т.е. в данном случае запрашивается **число**, а **не массив данных или не элемент этого массива**).

```
05 00
T  L )
```

- Вторая переменная в данной последовательности, означает NULL (тэг=05), что в данном случае означает алгоритмический 0, т.е. окончание первой переменной.

Аналогично расшифруем остальные переменные в сообщении №1.

```
30 0e 06 0a 2b 06 01 02 01 02 02 01 05 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 08 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 09 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 0a 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 0b 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 0c 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 0d 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 0e 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 10 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 11 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 12 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 13 01 05 00
30 0e 06 0a 2b 06 01 02 01 02 02 01 14 01 05 00
```

Во-первых, отметим, что все оставшиеся ИЭ имеют одинаковую длину (0e'hex или 14 байт) и представляют собой последовательности (тэг=30), каждая из которых содержит по две переменные:

- идентификатор объекта (OID – тэг=06, длина 0a'hex или по 10 байт) и
- алгоритмический конец переменной, обозначенный как NULL (тэг=05, длина=00, а содержимое - отсутствует).

Во-вторых, отметим, что все объекты, идентификаторы которых указаны в данном запросе – находятся по одному пути:

2b 06 01 02 01 02 02 01, который означает:

```
iso.org.dod.internet.mgmt.mib.if.ifTable.ifEntry, или
 1 . 3 . 6 .    1   . 2  . 1 .2 .   2    .  1
```

Для расшифровки элементов, относящихся к группе объектов if (интерфейс), необходимо обратиться документу RFC-1213 (MIB-II), который прилагается в электронном виде, и основные выдержки из которого приведены в приложении 2, а также (кратко) ниже.

Итак, группа объектов if , принадлежащая ветви MIB

iso.org.dod.internet.mgmt.mib.if

1 . 3 . 6 . 1 . 2 . 1 .2,

содержит 22 элемента (объекта управления), находящихся в подветви **if.ifTable.ifEntry (или 2.2.1).**

Более детальное описание этих объектов см. в приложении 2, а также в документе RFC-1213, здесь же кратко приводятся идентификаторы этих объектов:

Объекты if ::= { interfaces 2 } (RFC-1213)

```
        ifTable OBJECT-TYPE ::= { interfaces 2 }
             ifEntry OBJECT-TYPE ::= { ifTable 1 }
             IfEntry ::=  SEQUENCE {
                ifIndex           ::= { ifEntry 1 }
                ifDescr           ::= { ifEntry 2 }
                ifType            ::= { ifEntry 3 }
                ifMtu             ::= { ifEntry 4 }
                ifSpeed           ::= { ifEntry 5 }
                ifPhysAddress     ::= { ifEntry 6 }
                ifAdminStatus     ::= { ifEntry 7 }
                ifOperStatus      ::= { ifEntry 8 }
                ifLastChange      ::= { ifEntry 9 }
                ifInOctets        ::= { ifEntry 10 }
                ifInUcastPkts     ::= { ifEntry 11 }
                ifInNUcastPkts    ::= { ifEntry 12 }
                ifInDiscards      ::= { ifEntry 13 }
                ifInErrors        ::= { ifEntry 14 }
                ifInUnknownProtos ::= { ifEntry 15 }
                ifOutOctets       ::= { ifEntry 16 }
                ifOutUcastPkts    ::= { ifEntry 17 }
                ifOutNUcastPkts   ::= { ifEntry 18 }
                ifOutDiscards     ::= { ifEntry 19 }
                ifOutErrors       ::= { ifEntry 20 }
                ifOutQLen         ::= { ifEntry 21 }
                ifSpecific        ::= { ifEntry 22 }
            }
```

Теперь, обращаясь к трассировке сообщения №1, видим, что из 22-х возможных объектов, менеджера в нашем запросе интересуют следующие объекты (их идентификаторы приведены в конце пути **2b 06 01 02 01 02 02 01**):

```
05 01   - ifSpeed          ::= { ifEntry 5 }
08 01   - ifOperStatus     ::= { ifEntry 8 }
09 01   - ifLastChange     ::= { ifEntry 9 }
0a 01   - ifInOctets            ::= { ifEntry 10 }
0b 01   - ifInUcastPkts    ::= { ifEntry 11 }
0c 01   - ifInNUcastPkts        ::= { ifEntry 12 }
0d 01   - ifInDiscards     ::= { ifEntry 13 }
0e 01   - ifInErrors            ::= { ifEntry 14 }
10 01   - ifOutOctets      ::= { ifEntry 16 }
11 01   - ifOutUcastPkts        ::= { ifEntry 17 }
12 01   - ifOutNUcastPkts       ::= { ifEntry 18 }
13 01   - ifOutDiscards    ::= { ifEntry 19 }
14 01   - ifOutErrors      ::= { ifEntry 20 }
```

Единица (01’hex) в конце пути указывает на то, что запрашивается элемент массива из база данных.

В контрольной работе необходимо также привести текстовое описание каждого из запрашиваемых объектов.

ВНИМАНИЕ:

В приложении 2 и в документе RFC-1213 это описание приводится на английском языке. В ответах, помещаемых по прилагаемым в контрольной работе формам, необходимо это описание приводить на русском языке!

Ответим на оставшиеся вопросы (выделены ниже синим цветом) по сообщению №1 и впишем эти ответы в прилагаемые формы.

Определить из приведенных сообщений (см. выделенное синим цветом):

1. Фирму-поставщика оборудования сетевых интерфейсов
2. MAC-адреса источника и назначения
3. Тип протокола, обслуживаемого данным Ethernet-кадром
4. Версию протокола сетевого уровня
5. Приоритет сетевого уровня для данной дейтаграммы
6. Длину пакета сетевого уровня (в байтах)
7. Время жизни данной дейтаграммы
8. Протокол транспортного уровня (Dec'код и название)
9. Сетевой адрес отправителя
10. Сетевой адрес назначения
11. Транспортный порт отправителя
12. Транспортный порт получателя

13. Тип и версию протокола прикладного уровня
14. Длину дейтаграммы транспортного уровня (в байтах)
15. Тип и класс тэга протокола прикладного уровня
16. Длину сообщения протокола прикладного уровня
17. Длину и содержимое поля Community
18. Тип PDU и его длину (в байтах)
    - 18.1. Для PDU типа **Get-Request**
      - 18.1.1. Значение идентификатора запроса - **RequestID**
      - 18.1.2. Значения полей **ErrorStatus и Errorlndex**
      - 18.1.3. Длину поля, содержащего набор запрашиваемых характеристик
      - 18.1.4. Перечень запрашиваемых характеристик (атрибутов) управляемого объекта*
    - 18.2. Для PDU типа **GetResponse**
      - 18.2.1. Значение идентификатора запроса – **RequestID**
      - 18.2.2. Значения полей **ErrorStatus и Errorlndex**
      - 18.2.3. Длину поля, содержащего набор характеристик управляемого объекта
      - 18.2.4. Перечень характеристик (атрибутов) управляемого объекта*
      - 18.2.5. Значения характеристик (атрибутов) управляемого объекта*

№ вопроса	Для PDU типа **Get-Request**	
	Hex’ значение	Dec’ или текстовое значение
1	00 00 1d 00 20 af	– Сетевой интерфейс фирмы **Cabletron** – Сетевой интерфейс фирмы **3COM**
2	00 00 1d 90 58 20 00 20 af e8 e2 8e	MAC-адрес назначения MAC-адрес источника
3	08 00	протокол IPv4
4	4	4-я версия
5	00	000 – низкий приоритет
6	01 1a	282 байта
7	40	TTL=64 транзита
8	11	17 – UDP протокол
9	d4 a4 00 66	212.164.00.102
10	d4 a4 c4 f6	212.164.196.246
11	c0 7c	49351 - DP
12	00 a1	161 - SNMP
13	00 a1	161 - SNMP
14	01 06	262 байта
15	30	Класс UNI, тип составной, последовательность (Sequence)
16	81 fb	fb’hex или 251 байт
17	06 76 6d 31 35 2d 31	Поле “Community” длиной 6 байт, содержимое: vm15-1
18	a0 81 ed	– имя PDU-SNMP. В данном случае – это Get-request (тэг=a0). Длина содержимого в этом PDU составляет ed’hex или 237’dec байт
18.1.1	35 97 ac 55	899132501’dec
18.1.2	00	Оба поля имеют значение 00
18.1.3	81 de	222’dec байт – длина поля переменных

Для **Get-Request (вопрос** 18.1.4**):**

№		Наименование атрибута (OID)	Значение атрибута (характеристики)
1	Hex’	2b 06 01 02 01 01 03 00	Краткое описание объекта sysUpTime на русском языке (см. приложение 2 и RFC-1213) **Перевести на русский:** «The time (in hundredths of a second) since the network management portion of the system was last reinitialized»
	Dec’	1.3.6.1.2.1.1.3.0	
	Текст	iso.org.dod.internet.mgmt.mib.sys.sysUpTime.0	
2	Hex’	**2b 06 01 02 01 02 02 01 05 01**	Краткое описание объекта **ifSpeed** на русском языке (см. приложение 2 и RFC-1213)
	Dec’	1.3.6.1.2.1.2.2.1.5.1	
	Текст	**ifSpeed**	
3	Hex’	**2b 06 01 02 01 02 02 01 08 01**	Краткое описание объекта **ifOperStatus** на русском языке (см. приложение 2 и RFC-1213)
	Dec’	1.3.6.1.2.1.2.2.1.8.1	
	Текст	**ifOperStatus**	
4	Hex’	**2b 06 01 02 01 02 02 01 09 01**	Краткое описание объекта **ifLastChange** на русском языке (см. приложение 2 и RFC-1213)
	Dec’	1.3.6.1.2.1.2.2.1.9.1	
	Текст	**ifLastChange**	
5	Hex’	**2b 06 01 02 01 02 02 01 0a 01**	-//-//-//-
	Dec’	1.3.6.1.2.1.2.2.1.10.1	
	Текст	**ifInOctets**	
	… и т.д. (т.е. заполнить всю форму)		

Расшифруем теперь сообщение №2

2. Сообщение №2

```
0000: 00 20 af e8 e2 8e 00 00   1d 7c 63 f1 08 00 45 00
0010: 01 37 9c bf 00 00 3e 11   70 56 d4 a4 c4 f1 d4 a4
0020: 00 66 00 a1 c0 7a 01 23   7b 84 30 82 01 17 02 01
0030: 00 04 05 65 78 70 2d 31   a2 82 01 09 02 04 35 97
0040: ac 59 02 01 00 02 01 00   30 81 fa 30 0f 06 08 2b
0050: 06 01 02 01 01 03 00 43   03 73 d4 70 30 11 06 0a
0060: 2b 06 01 02 01 02 02 01   05 03 42 03 00 fa 00 30
0070: 0f 06 0a 2b 06 01 02 01   02 02 01 08 03 02 01 01
0080: 30 0f 06 0a 2b 06 01 02   01 02 02 01 09 03 43 01
0090: 00 30 12 06 0a 2b 06 01   02 01 02 02 01 0a 03 41
00a0: 04 04 12 5a 5d 30 11 06   0a 2b 06 01 02 01 02 02
00b0: 01 0b 03 41 03 08 6f da   30 0f 06 0a 2b 06 01 02
00c0: 01 02 02 01 0c 03 41 01   07 30 0f 06 0a 2b 06 01
00d0: 02 01 02 02 01 0d 03 41   01 00 30 0f 06 0a 2b 06
00e0: 01 02 01 02 02 01 0e 03   41 01 00 30 12 06 0a 2b
00f0: 06 01 02 01 02 02 01 10   03 41 04 13 a1 03 ca 30
0100: 11 06 0a 2b 06 01 02 01   02 02 01 11 03 41 03 08
0110: 0d 32 30 0f 06 0a 2b 06   01 02 01 02 02 01 12 03
0120: 41 01 00 30 0f 06 0a 2b   06 01 02 01 02 02 01 13
0130: 03 41 01 00 30 0f 06 0a   2b 06 01 02 01 02 02 01
0140: 14 03 41 01 00
```

Все поля, относящиеся к заголовкам протоколов Ethernet, IP, UDP расшифруем аналогично и сразу заполним в предлагаемую форму, тем самым ответим на первые 14 вопросов.

№ вопроса	Для PDU типа **Get-Request**		Для PDU типа **Get-Response**	
	Hex' значение	Dec' или текстовое значение	Hex' значение	Dec' или текстовое значение
1	00 00 1d 00 20 af	– Сетевой интерфейс фирмы **Cabletron** – Сетевой интерфейс фирмы **3COM**	00 00 1d 00 20 af	– Сетевой интерфейс фирмы **Cabletron** – Сетевой интерфейс фирмы **3COM**
2	00 00 1d 90 58 20 00 20 af e8 e2 8e	MAC-адрес назначения MAC-адрес источника	00 20 af e8 e2 8e 00 00 1d 90 58 20	MAC-адрес назначения MAC-адрес источника
3	08 00	протокол IPv4	08 00	протокол IPv4
4	4	4-я версия	4	4-я версия
5	00	000 – низкий приоритет	00	000 – низкий приоритет
6	01 1a	282 байта	01 37	311 байт
7	40	TTL=64 транзита	3e	TTL=62 транзита
8	11	17 – UDP протокол	11	17 – UDP протокол
9	d4 a4 00 66	212.164.00.102	d4 a4 c4 f6	212.164.196.246
10	d4 a4 c4 f6	212.164.196.246	d4 a4 00 66	212.164.00.102
11	c0 7c	49351 - DP	c0 7c	49351 - DP
12	00 a1	161 - SNMP	00 a1	161 - SNMP
13	00 a1	161 - SNMP	00 a1	161 - SNMP
14	01 06	262 байта	01 23	291 байт

Аналогично будем действовать и при расшифровке полей сообщения, относящихся к PDU-SNMP-Response, имея при этом ввиду, что в ответном сообщении агент передает **значения** запрашиваемых параметров.

Сначала выпишем отдельно эти поля:

```
0020:                                           30 82 01 17 02 01
0030: 00 04 05 65 78 70 2d 31   a2 82 01 09 02 04 35 97
0040: ac 59 02 01 00 02 01 00   30 81 fa 30 0f 06 08 2b
0050: 06 01 02 01 01 03 00 43   03 73 d4 70 30 11 06 0a
0060: 2b 06 01 02 01 02 02 01   05 03 42 03 00 fa 00 30
0070: 0f 06 0a 2b 06 01 02 01   02 02 01 08 03 02 01 01
0080: 30 0f 06 0a 2b 06 01 02   01 02 02 01 09 03 43 01
0090: 00 30 12 06 0a 2b 06 01   02 01 02 02 01 0a 03 41
00a0: 04 04 12 5a 5d 30 11 06   0a 2b 06 01 02 01 02 02
00b0: 01 0b 03 41 03 08 6f da   30 0f 06 0a 2b 06 01 02
00c0: 01 02 02 01 0c 03 41 01   07 30 0f 06 0a 2b 06 01
00d0: 02 01 02 02 01 0d 03 41   01 00 30 0f 06 0a 2b 06
00e0: 01 02 01 02 02 01 0e 03   41 01 00 30 12 06 0a 2b
00f0: 06 01 02 01 02 02 01 10   03 41 04 13 a1 03 ca 30
0100: 11 06 0a 2b 06 01 02 01   02 02 01 11 03 41 03 08
0110: 0d 32 30 0f 06 0a 2b 06   01 02 01 02 02 01 12 03
0120: 41 01 00 30 0f 06 0a 2b   06 01 02 01 02 02 01 13
0130: 03 41 01 00 30 0f 06 0a   2b 06 01 02 01 02 02 01
0140: 14 03 41 01 00
```

Выделим теперь составные части этого сообщения, пользуясь общим форматом SNMP и форматом PDU-Response:

```
30 82 01 17
02 01 00
04 05 65 78 70 2d 31
a2 82 01 09
02 04 35 97 ac 59
02 01 00
02 01 00
30 81 fa
30 0f 06 08 2b 06 01 02 01 01 03 00 43 03 73 d4 70
30 11 06 0a 2b 06 01 02 01 02 02 01 05 03 42 03 00 fa 00
30 0f 06 0a 2b 06 01 02 01 02 02 01 08 03 02 01 01
30 0f 06 0a 2b 06 01 02 01 02 02 01 09 03 43 01 00
30 12 06 0a 2b 06 01 02 01 02 02 01 0a 03 41 04 04 12 5a 5d
30 11 06 0a 2b 06 01 02 01 02 02 01 0b 03 41 03 08 6f da
30 0f 06 0a 2b 06 01 02 01 02 02 01 0c 03 41 01 07
30 0f 06 0a 2b 06 01 02 01 02 02 01 0d 03 41 01 00
30 0f 06 0a 2b 06 01 02 01 02 02 01 0e 03 41 01 00
30 12 06 0a 2b 06 01 02 01 02 02 01 10 03 41 04 13 a1 03 ca
30 11 06 0a 2b 06 01 02 01 02 02 01 11 03 41 03 08 0d 32
30 0f 06 0a 2b 06 01 02 01 02 02 01 12 03 41 01 00
30 0f 06 0a 2b 06 01 02 01 02 02 01 13 03 41 01 00
30 0f 06 0a 2b 06 01 02 01 02 02 01 14 03 41 01 00
```

Здесь подчеркнуты OID, запрашиваемых объектов, согласно RFC-1213.

В отличие от сообщения-запроса в данном сообщении-ответа в конце идентификаторов объектов приводится значение 03. Это означает, что передается элемент из массив данных (таблицы).

Необходимо также определить тип данных этих элементов.

Пользуясь указаниями п.2.3.5.1 и RFC-1213, определяем, что передаваемые в ответе ИЭ имеют следующие типы (см. соответствующие тэги):

- Тэг=43 – тип данных составной из класса «Прикладной» (01), код тэга 3 означает, что это TimeTicks.
- Тэг=42 – тип данных составной из класса «Прикладной» (01), код тэга 2 означает, что это Gauge (величина).
- Тэг=41 – тип данных составной из класса «Прикладной» (01), код тэга 1 означает, что это Counter (счетчик).
- Тэг=02 – тип данных составной из класса «UNI» (00), код тэга 2 означает, что это Integer (целое).

Учитывая сказанное по сообщению №2, ответим на оставшиеся вопросы по этому сообщению и впишем эти ответы в прилагаемые формы.

Определить из приведенных сообщений:

1. Фирму-поставщика оборудования сетевых интерфейсов
2. MAC-адреса источника и назначения
3. Тип протокола, обслуживаемого данным Ethernet-кадром
4. Версию протокола сетевого уровня
5. Приоритет сетевого уровня для данной дейтаграммы
6. Длину пакета сетевого уровня (в байтах)
7. Время жизни данной дейтаграммы
8. Протокол транспортного уровня (Dec'код и название)
9. Сетевой адрес отправителя
10. Сетевой адрес назначения
11. Транспортный порт отправителя
12. Транспортный порт получателя
13. Тип и версию протокола прикладного уровня
14. Длину дейтаграммы транспортного уровня (в байтах)
15. Тип и класс тэга протокола прикладного уровня
16. Длину сообщения протокола прикладного уровня
17. Длину и содержимое поля Community
18. Тип PDU и его длину (в байтах)
    - 18.1. Для PDU типа **Get-Request**
      - 18.1.1. Значение идентификатора запроса - **RequestID**
      - 18.1.2. Значения полей **ErrorStatus и Errorlndex**
      - 18.1.3. Длину поля, содержащего набор запрашиваемых характеристик
      - 18.1.4. Перечень запрашиваемых характеристик (атрибутов) управляемого объекта*
    - 18.2. Для PDU типа **GetResponse**
      - 18.2.1. Значение идентификатора запроса – **RequestID**
      - 18.2.2. Значения полей **ErrorStatus и Errorlndex**
      - 18.2.3. Длину поля, содержащего набор характеристик управляемого объекта
      - 18.2.4. Перечень характеристик (атрибутов) управляемого объекта*
      - 18.2.5. Значения характеристик (атрибутов) управляемого объекта*

№ вопроса	Для PDU типа **Get-Response**	
	Hex'значение	Dec' или текстовое значение
1	00 00 1d 00 20 af	– Сетевой интерфейс фирмы **Cabletron** – Сетевой интерфейс фирмы **3COM**
2	00 20 af e8 e2 8e 00 00 1d 90 58 20	MAC-адрес назначения MAC-адрес источника
3	08 00	протокол IPv4
4	4	4-я версия
5	00	000 – низкий приоритет
6	01 37	311 байт
7	3e	TTL=62 транзита
8	11	17 – UDP протокол
9	d4 a4 c4 f6	212.164.196.246
10	d4 a4 00 66	212.164.00.102
11	c0 7c	49351 - DP
12	00 a1	161 - SNMP
13	00 a1	161 - SNMP
14	01 23	291 байт
15	30	Класс UNI, тип составной, последовательность (Sequence)
16	81 fb	fb'hex или 251 байт
17	05 65 78 70 2d 31	Поле "Community" длиной 5 байт, содержимое: exp-1
18	a2 82 01 09	– имя PDU-SNMP. В данном случае – это Get- **Response** (тэг=a2). Длина содержимого в этом PDU составляет **01 09**'hex или 265'dec байт
18.2.1	35 97 ac 59	899132505'dec
18.2.2	00	Оба поля имеют значение 00
18.2.3	81 **fa**	250'dec байт – длина поля переменных

Для Get- **Response (вопрос** 18.2.4 и 18.2.5**):**

№		Наименование атрибута (OID)	Значение атрибута (характеристики)
1	Hex'	2b 06 01 02 01 01 03 00	**73 d4 70**
	Dec'	1.3.6.1.2.1.1.3.0	7591024*100c
	Текст	iso.org.dod.internet.mgmt.mib.sys.sysUpTime.0	
2	Hex'	**2b 06 01 02 01 02 02 01 05 01**	**fa 00**
	Dec'	1.3.6.1.2.1.2.2.1.5.1	64 000 бит/с
	Текст	**ifSpeed**	
3	Hex'	**2b 06 01 02 01 02 02 01 08 01**	01
	Dec'	1.3.6.1.2.1.2.2.1.8.1	1
	Текст	**ifOperStatus**	Оперативное состояние объекта - up
4	Hex'	**2b 06 01 02 01 02 02 01 09 01**	00
	Dec'	1.3.6.1.2.1.2.2.1.9.1	0
	Текст	**ifLastChange**	
5	Hex'	**2b 06 01 02 01 02 02 01 0a 01**	-//-//-//-
	Dec'	1.3.6.1.2.1.2.2.1.10.1	
	Текст	**ifInOctets**	
	… и т.д. (т.е. заполнить всю форму)		

На этом первую часть задания можно считать выполненным.

Необходимо оформить его в соответствии с требованиями п.2.6.

2.6 Правила оформления первой части работы:

1. На титульном листе привести следующие данные:
   1.1. Название дисциплины
   1.2. Название работы
   1.3. ФИО
   1.4. № группы
   1.5. № варианта
2. На первой странице работы обязательно привести Ваш вариант задания (шестнадцатиричные значения сообщений управляющего протокола)
3. На следующих страницах привести ответы на поставленные вопросы
4. Ответы оформить в виде таблицы, следующего вида:

№ вопроса	Для PDU типа **Get-Request**		Для PDU типа **Get-Response**	
	Hex' значение	Dec' или текстовое значение	Hex' значение	Dec' или текстовое значение
1				
2				
…				
18.1.1				
18.1.2				
…				
18.2.1				
18.2.2				
…				

5. Ответы на вопросы, отмеченные (*) свести в таблицы следующего вида:

Для **Get-Request (вопрос** 18.1.4**):**

№		Наименование атрибута (OID)	Значение атрибута (характеристики)
1	Hex'		
	Dec'		
	Текст		
2	Hex'		
	Dec'		
	Текст		
…			

Для **Get-Response (вопросы** 18.2.4 и 18.2.5**):**

№		Наименование атрибута (OID)	Значение атрибута (характеристики)
1	Hex'		
	Dec'		
	Текст		
2	Hex'		
	Dec'		
	Текст		
…			

2.7 Выводы по первой части контрольной работы:

В данной части работы, посвященной особенностям функционирования управляющей платформы, выполнено следующее:
- рассмотрена схема взаимодействия Менеджер-Агент
- рассмотрены стеки протоколов для организации этого взаимодействия
- расшифрованы заголовки транспортных протоколов
- рассмотрена структура протокола управления SNMP
- рассмотрена структура и состав данных MIB на примере MIB RFC 1213
- рассмотрен язык описания данных MIB и протокола SNMP (ASN.1)
- рассмотрены правила кодирования T-L-V

3 Проект ресурсов транспортной сети (2-я часть контрольной работы)

Для проектирования транспортных ресурсов мультисервисной сети (МСС), выделяемых под обмен управляющей информацией, необходимо знать следующие данные об этой управляющей информации и ее источниках:

- Количество менеджеров (в данном проекте – задан один менеджер)
- Количество агентов (связано с количеством управляемых объектов и задается в каждом варианте)
- Характеристики потока запросов/ответов менеджер-агент
  - Средняя скорость (интенсивность) передачи информации
  - Кпач от агента к менеджеру
  - Пиковая скорость (интенсивность) передачи информации
  - Параметры качества, характеризующие пропуск управляющего трафика (CoS)

Задание на 2-ю часть контрольной работы

Исходные данные:

1. Модель системы обслуживания с одним менеджером
2. Количество агентов и их характеристики

В табл. 3.1 заданы варианты количества агентов, размещенных на управляемых объектах, а также характеристики агентов.

В данном проекте **рассчитываем только трафик от агентов к менеджеру**, полагая, что трафик от менеджера к агенту незначительный, детерминированный и определяется в основном запросами менеджера по заранее составленному расписанию, либо запросами менеджера, вводимыми вручную операторами ОМС.

В данном проекте интенсивность трафика зададим параметрами скорости передачи управляющей информации – Vср и Кпачечности.

Тип трафика **от агентов к менеджеру полагаем псевдослучайным**, т.к. в нем имеется явно выраженная **детерминированная составляющая** (ответы агента менеджеру по расписанию, но объемы ответов – случайные величины), а также **случайные составляющие**, интенсивность которых, зависит от:

- вероятности отказов на управляемых агентах (сообщения протокола Trap),
- объема статистической, биллинговой и другой информации, собираемой с управляемых объектов. Сюда может входить управляющая информация всех типов – FM, PM, CM, AM, SM.

Таблица 3.1 – Количество агентов и параметры управляющего трафика

n/n вариант	Nагент	Vср (от каждого агента) Кбит/с	Кпач
1	500	3	3
2	600	4,2	5
3	200	8,5	12,7
4	900	2	25
5	1400	1,5	5
6	2300	2,8	4
7	800	12	15,8
8	1200	7	21
9	1500	4,8	28
10	400	3,2	16
11	2100	8,5	6
12	2200	2	7
13	700	1,5	14
14	850	2,8	23,6
15	1340	12	17
16	2270	12	4
17	2300	7	15,8
18	800	4,8	21
19	1200	3,2	28
20	1500	8,5	16
21	400	12	6
22	1500	7	7
23	400	4,8	14
24	2100	3,2	23,6
25	2200	8,5	5
26	700	12	4
27	900	11	15,8
28	1400	6,5	21
29	2300	5,4	28
30	800	7,7	11,3

Здесь Nагент – количество агентов на управляемых объектах для каждого варианта.

Требуется рассчитать и обосновать:

Вначале этой части проекта необходимо разработать структурную схему управляемой сети с обозначением на ней управляемых объектов (наличие ПО Агента и MIB) и менеджеров (серверов, собирающих управляющую информацию от агентов).

В зону проектирования данного проекта входят только элементы IP-сети (маршрутизаторы, коммутаторы, шлюзы, серверы и т.п.). Объекты, не имеющие интерфейсов IP/Ethernet, в данном проекте не рассматриваются.

Также необходимо представить структурную схему серверного узла – менеджера, реализованного в центре эксплуатации и ТО сети – ОМС. Расчет количества операторских мест в ОМС не входит в задачи данного проекта.

Для всех типов управляющего трафика создается корпоративная виртуальная сеть – VPN, для которой необходимо определить следующие параметры:

1. Задать номер виртуальной сети (соответствует номеру варианта + 100)
2. Рассчитать требуемую пропускную способность от всех агентов к менеджеру в интерфейсе между SW1-SW3 (рис. 3.2).
3. Задать класс обслуживания трафика (CoS или приоритет управляющего трафика). В данном проекте зададимся значением CoS = 4 (или 100’bin)

3.1 Разработка структурной схемы проектируемой сети

На рис.3.1 представлена обобщенная схема управляемой сети, при этом элементы транспортной сети (маршрутизаторы, коммутаторы) одновременно являются объектами управления и они же обеспечивают транспортный ресурс (пропускную способность) для обмена управляющей информацией между центральным менеджером (ОМС) и агентским ПО, расположенным на этом транспортном /управляемом элементе.

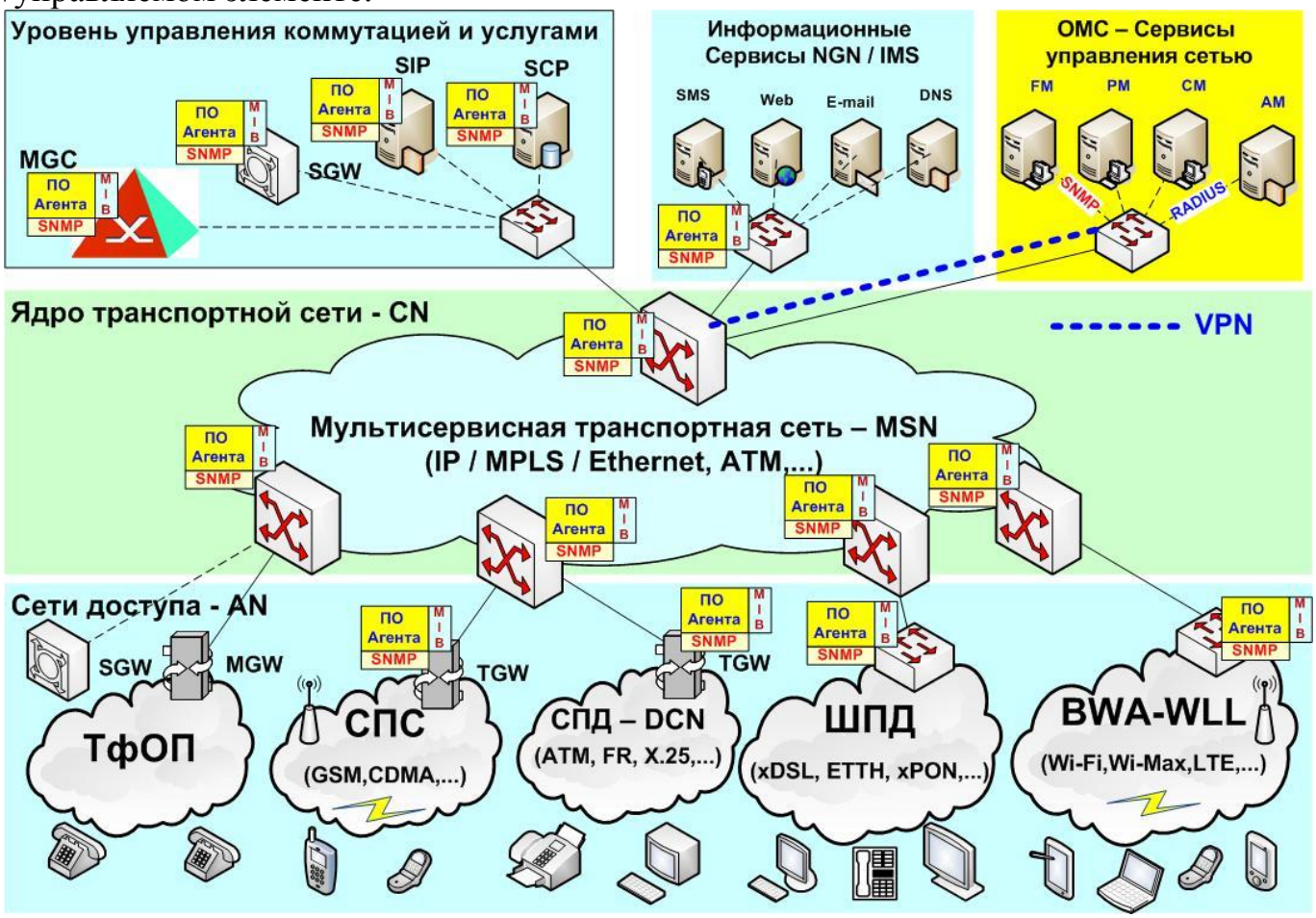


Рисунок 3.1 – Структурная схема управляемой сети

Для обмена управляющей информацией между агентами и менеджером необходимо организовать виртуальную частную сеть (VPN). На рис.3.1 и 3.2 эта сеть условно обозначена штриховой синей линией.

Структурная схема серверного узла (менеджера)

Центр эксплуатации и ТО сети – OMC представляет собой набор серверов, выполняющих функции менеджеров – см. желтый квадрат справа вверху на рис.3.1.

На рис. 3.2 OMC показан более детально. Управляющие серверы условно обозначены как FM, PM, CM, AM, SM, т.е. по основным выполняемым функциям – управление отказами, статистикой, конфигурацией, биллингом, безопасностью, выполняющих функции менеджеров.

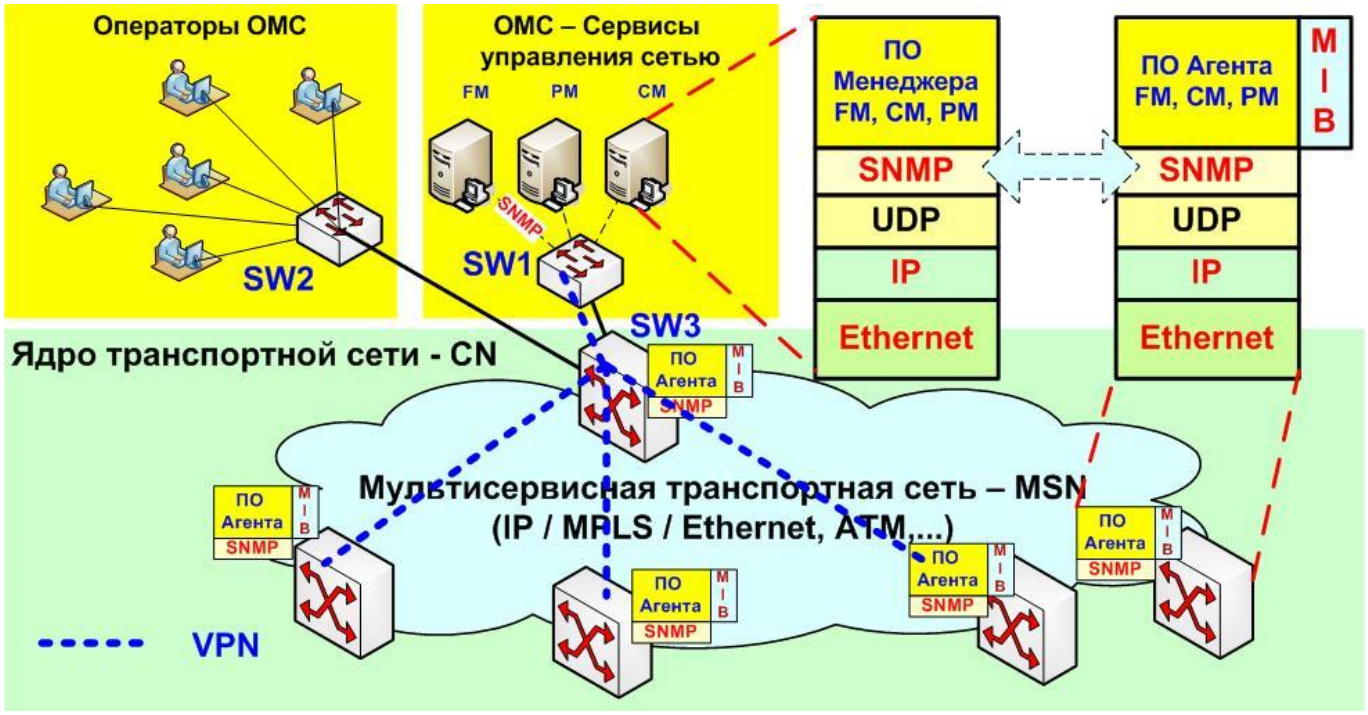


Рисунок 3.2 – Структурная схема серверного узла (менеджера – OMC)

Вопросы размещения ПО менеджера на одном аппаратном сервере или на нескольких решаются путем оценки требуемой производительности менеджера, зависящей от количества управляемых объектов (агентов), т.е. от масштаба и размеров управляемой сети.

В данном проекте оценка требуемой производительности не выполняется, но предполагается, что каждый менеджер размещен на отдельном сервере, а эти серверы связаны в локальной корпоративной сети через коммутатор SW1, который в свою очередь имеет доступ к публичной управляемой сети (коммутатор SW3).

В интерфейсе SW1-SW3 (рис. 3.2) поддерживаются функции обеспечения информационной безопасности, благодаря корпоративной VPN для сервисов управления (обозначена жирной пунктирной синей линией).

3.2 Расчет нагрузки на транспортную сеть от агентов и менеджеров

Рассчитаем нагрузку от всех агентов на одного менеджера в самом «узком» месте схемы рис. 3.2 – а именно в интерфейсе SW3-SW1.

Для трафика не реального времени, можно использовать статистическое мультиплексирование, т.е. суммировать трафик от разных услуг и агентов с учетом $К_{пач}$. В данном проекте полагаем, что значение коэффициента пачечности для всех услуг и агентов одинаково и задано в табл. 3.1.

Пусть заданы следующие исходные данные:

n/n вариант	Nагент	Vср (от каждого агента) Кбит/с	Кпач
55	1000	4	5

Расчет ведем на максимальную (пиковую) нагрузку, для чего определим значение пиковой скорости в интерфейсе SW1-SW3:

Vпик = Vср * Nагент * Кпач = 4 * 1000 * 5 = 20 000 Кбит/с = 20 Мбит/с

3.3 Расчет пропускной способности сетевых интерфейсов

Весь трафик от агентов к менеджеру будет пропускаться через интерфейс SW3-SW1, в котором необходимо предусмотреть пропускную способность:

C (кбит/с) = Vпик = 20 Мбит/с

3.4 Выбор ПО менеджера (системы управления/мониторинга)

В этом пункте необходимо выбрать ПО менеджера (систему управления/мониторинга) и кратко описать основные характеристики этой системы.

Для выбора ПО менеджера можно ориентироваться на популярные системы типа Nagios, Zabbix, Cacti, HP-OV и т.п. – см. ресурсы в сети Интернет [6…10].

В качестве параметров, влияющих на выбор, могут рассматриваться следующие характеристики:

- масштаб сети (количество агентов)
- значимость бренда (торговой марки)
- лицензированное или свободное ПО
- тип ОС, служащей платформой для ПО менеджера
- стоимость ПО менеджера и другие характеристики.

3.5 Конфигурация VPN для организации сети управления

VPN (ВЧС – виртуальная частная сеть) – услуга предоставления выделенных логических (виртуальных) ресурсов в публичной физической сети для целей безопасности частного трафика путем его изоляции от других клиентов.

VPN – это виртуальная надстройка над физической локальной сетью (LAN) или сетью уровня метро (MAN), которая может быть организована средствами следующих технологий:

- ATM – за счет адресов VPI в общей части адреса уровня L2 – VPI/VCI
- MPLS – за счет части 20-ти битовой метки Label из общего 32-х битного пространства
- IEEE 802.1p/Q – за счет 12-ти битовой метки VID из общего 16-ти битового пространства (за ней закрепилось название VLAN)

В данном проекте мы рассматриваем технологию поддержки VPN для управляющего трафика на основе стандарта IEEE 802.1 p/Q (VLAN), который допускает поддержку до 4096 виртуальных сетей, организованных по клиентам (с привязкой к портам коммутатора и MAC-адресам), либо по типу сервиса.

Протокол IEEE 802.1Q – это открытый стандарт, который описывает процедуру тегирования (разметки) трафика для передачи информации о принадлежности к VLAN.

Сегментирование сети на отдельные VLAN-подсети возможно благодаря специальному 12-битовому полю VID в заголовке кадра Ethernet (см. рис.3.3 - Формат кадра).

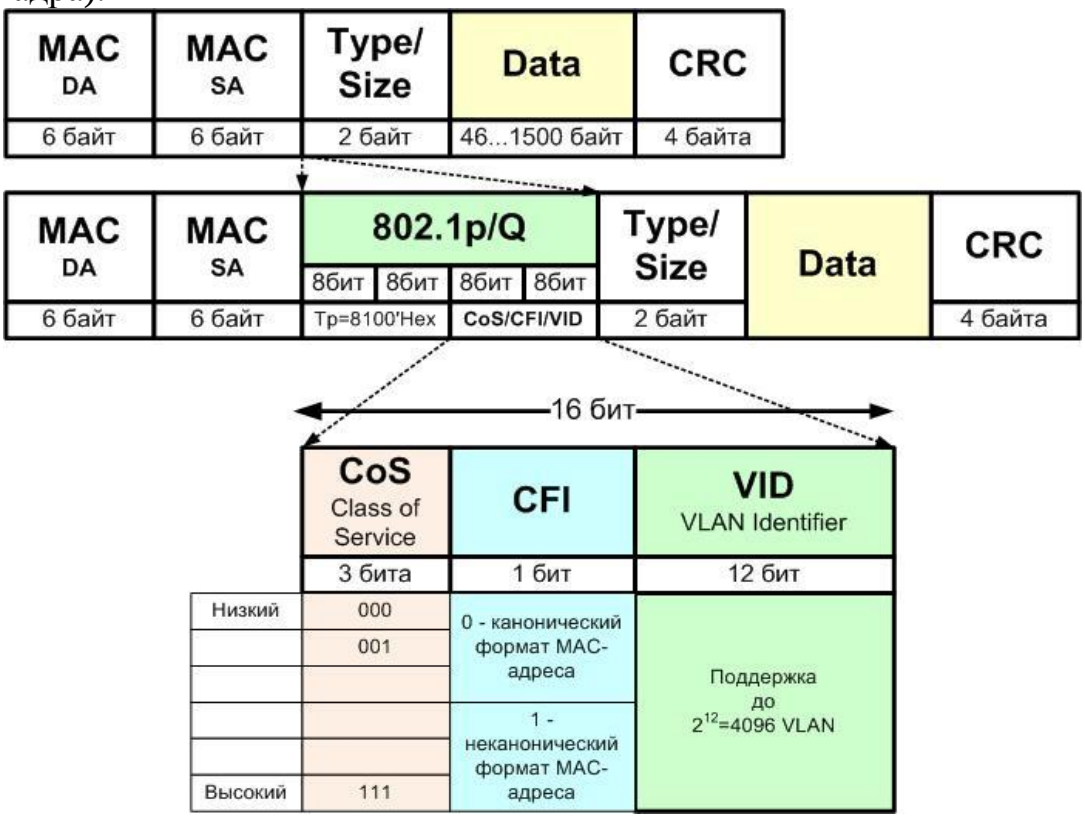


Рисунок 3.3 – Формат кадра Ethernet с тегом 802.1p/Q

Протокол **802.1Q** помещает внутрь кадра Ethernet **тег (метку)**, значение которого информирует Ethernet-switch о принадлежности трафика к конкретной VLAN.

Размер тега – 4 байта. Он состоит из таких полей:

- TPID – Tag Protocol Identifier, идентификатор протокола тегирования. Размер поля – 16 бит. Указывает на то, какой протокол используется для разметки кадров. Для 802.1Q зарезервировано значение TPID=8100'Hex.
- CoS – Class of Services, класс сервиса – используется **протоколом 802.1p** для задания приоритета передаваемого трафика. Размер поля – 3 бита..
- CFI – Canonical Format Indicator, индикатор канонического формата. Размер поля — 1 бит. Указывает на формат MAC-адреса. 0 – канонический, 1 – не канонический. CFI используется для совместимости между сетями Ethernet и Token Ring.
- VID – VLAN Identifier, идентификатор VLAN. Размер поля – 12 бит. Указывает на то, какому VLAN принадлежит фрейм. Диапазон возможных значений от 0 до 4094.

При использовании стандарта Ethernet II, протокол 802.1Q вставляет тег перед полем «Тип протокола». Так как фрейм изменился, пересчитывается контрольная сумма.

В стандарте 802.1Q существует понятие **Native VLAN**.

По умолчанию это **VLAN 1**. Трафик, передающийся в этом VLAN, не тегируется.

Значение поля VID определяет принадлежность пакета к конкретной виртуальной подсети – VLAN. Метки, или теги (tags), иногда называют "VLAN-теги" или "Q-теги".

Таким образом, трафик, размеченный тегом конкретной VLAN, будет доступен только авторизованным абонентам, являющимся членами той же группы VLAN.

Зададим параметры VLAN, согласно параметрам на рис. 3.3, а также рассчитанной пропускной способности (задается при организации транка между SW1-SW3):

Номер VPN	VID	CoS	BW (кбит/с)
100 + Nварианта	**155**	**4**	**20 000**

3.5.1 Определение класса обслуживаемого трафика

В нашем варианте для сервиса управления задаем значение CoS = 4, хотя для отдельных типов управляющего трафика могут использоваться и значения наивысшие значения приоритета (например, для некоторых команд оператора, вводимых вручную).

3.6 ВЫВОДЫ

Основные выводы по данному проекту!!!

4. Литература

1. **Материалы сайта www.aek-54.ru (раздел УСС - http://www.aek-54.ru/tmn/tmn.htm)**
2. **Костюкович А.Е., Методические указания по выполнению контрольной работы по дисциплине ЭиТО СПС. 2012г. (прилагаются в электронном виде)**
3. **Гребешков А.Ю. Стандарты и технологии управления сетями связи. – М.:Эко-Трендз, 2003. – 288 с.**
4. **Олифер В.Г., Олифер Н.А., Компьютерные сети, Изд.3-е. СПб.: Питер, 2008**
5. **Шувалов В.П. и др. Телекоммуникационные системы, том 3, 2005.**
6. **Иванов А.Б., Засецкий А.В., Постников С.Д., Соколов И.В., Контроль качества в телекоммуникациях и связи. Часть 2, ,. М.: Syrus, 2001.**
7. РД 45.196.2001, "Правила построения системы телефонной связи общего пользования ".
8. Общее руководство качеством и элементы системы качества. - Международный стандарт ISO 9004–2, 1992.
9. ОТТ «Автоматизированные системы расчетов с пользователями за услуги электросвязи»
10. ГОСТ Р ИСО/МЭК 15408-1-2002, Информационные технологии. Методы и средства обеспечения безопасности, ч.1. Введение и общая модель. М.: Госстандарт России, 2002.
11. http://www.citforum.ru , Управление корпоративными ресурсами, Основные принципы стандартов семейства MRP-ERP. Эволюция систем корпоративного планирования
12. http://www.iso9000.ru , Организационные структуры предприятий.
13. Правила технической эксплуатации цифровых междугородных и международных телефонных станций сети электросвязи общего пользования Российской Федерации, М.,1998
14. МСЭ-Т, Управление сетью электросвязи. Рекомендации серии М.3000...М.3600, 1992 г.
15. IETF, RFC-1155
16. IETF, RFC-1157
17. IETF, RFC-1213

5. Приложения

Используемые сокращения

ИЭ – информационный элемент
СУ – система управления
УО – управляемый объект

SNMP - Simple Network Mangement Protocol – простой протокол управления сетью
CMIP - Common Management Information Protocol
MIB - Management Information Base - база данных управляющей информации
ASN.1 - Abstract Syntax Notation One - абстрактная синтаксическая нотация №1
BER - Basic Encoding Rules – базовые правила кодирования
IETF - Internet Engineering Task Force
ITU-T – International Telecommunication Union – Международный союз электросвязи (МСЭ-Т)
NMS – Network Management System – Система управления сетью
RFC – Request for Comments – Документы (стандарты) IETF для сети Интернет
PDU – Protocol Data Unit – Протокольный блок данных
OID – Object Identifier – Идентификатор объекта
MSB – Most Significant Bit – Наиболее значащий бит
LSB – Least Significant Bit – Наименее значащий бит

Приложение 1:

Объекты из группы interface (см. RFC-1213).

The Interfaces Group

```
   The definition of the ifNumber object was incorrect, as it required all interfaces
to support IP.  (For example, devices without IP, such as MAC-layer bridges, could
not be managed if this definition was strictly followed.)  The description of the
ifNumber object is changed accordingly.
   The ifTable object was mistaken marked as read-write, it has been (correctly) re-
designated as not-accessible.  In addition, several new values have been added to the
ifType column in the ifTable object:
      ppp(23)
      softwareLoopback(24)
      eon(25)
      ethernet-3Mbit(26)
      nsip(27)
      slip(28)
      ultra(29)
      ds3(30)
      sip(31)
      frame-relay(32)
   Finally, a new column has been added to the ifTable object:
      ifSpecific
   which provides information about information specific to the media
   being used to realize the interface.
          -- the Interfaces group
          -- Implementation of the Interfaces group is mandatory for all systems.

          ifNumber OBJECT-TYPE
              SYNTAX  INTEGER
              ACCESS  read-only
              STATUS  mandatory
              DESCRIPTION
                      "The number of network interfaces (regardless of
```

```
                    their current state) present on this system."
            ::= { interfaces 1 }

        -- the Interfaces table
        -- The Interfaces table contains information on the entity's
        -- interfaces.  Each interface is thought of as being
        -- attached to a `subnetwork'.  Note that this term should
        -- not be confused with `subnet' which refers to an
        -- addressing partitioning scheme used in the Internet suite
        -- of protocols.

        ifTable OBJECT-TYPE
            SYNTAX  SEQUENCE OF IfEntry
            ACCESS  not-accessible
            STATUS  mandatory
            DESCRIPTION
                    "A list of interface entries.  The number of
                    entries is given by the value of ifNumber."
            ::= { interfaces 2 }

        ifEntry OBJECT-TYPE
            SYNTAX  IfEntry
            ACCESS  not-accessible
            STATUS  mandatory
            DESCRIPTION
                    "An interface entry containing objects at the
                    subnetwork layer and below for a particular
                    interface."
            INDEX   { ifIndex }
            ::= { ifTable 1 }

        IfEntry ::=
            SEQUENCE {
                ifIndex          ::= { ifEntry 1 }
                    INTEGER,
                ifDescr          ::= { ifEntry 2 }
                    DisplayString,
                ifType           ::= { ifEntry 3 }
                    INTEGER,
                ifMtu            ::= { ifEntry 4 }
                    INTEGER,
                ifSpeed          ::= { ifEntry 5 }
                    Gauge,
                ifPhysAddress    ::= { ifEntry 6 }
                    PhysAddress,
                ifAdminStatus    ::= { ifEntry 7 }
                    INTEGER,
                ifOperStatus     ::= { ifEntry 8 }
                    INTEGER,
                ifLastChange     ::= { ifEntry 9 }
                    TimeTicks,
                ifInOctets       ::= { ifEntry 10 }
                    Counter,
                ifInUcastPkts    ::= { ifEntry 11 }
                    Counter,
                ifInNUcastPkts   ::= { ifEntry 12 }
                    Counter,
                ifInDiscards     ::= { ifEntry 13 }
                    Counter,
                ifInErrors       ::= { ifEntry 14 }
                    Counter,
                ifInUnknownProtos ::= { ifEntry 15 }
                    Counter,
                ifOutOctets      ::= { ifEntry 16 }
```

```
            Counter,
        ifOutUcastPkts    ::= { ifEntry 17 }
            Counter,
        ifOutNUcastPkts   ::= { ifEntry 18 }
            Counter,
        ifOutDiscards     ::= { ifEntry 19 }
            Counter,
        ifOutErrors       ::= { ifEntry 20 }
            Counter,
        ifOutQLen         ::= { ifEntry 21 }
            Gauge,
        ifSpecific        ::= { ifEntry 22 }
            OBJECT IDENTIFIER
    }

ifIndex OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "A unique value for each interface.  Its value
            ranges between 1 and the value of ifNumber.  The
            value for each interface must remain constant at
            least from one re-initialization of the entity's
            network management system to the next re-
            initialization."
    ::= { ifEntry 1 }

ifDescr OBJECT-TYPE
    SYNTAX  DisplayString (SIZE (0..255))
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "A textual string containing information about the
            interface.  This string should include the name of
            the manufacturer, the product name and the version
            of the hardware interface."
    ::= { ifEntry 2 }

ifType OBJECT-TYPE
    SYNTAX  INTEGER {
                other(1),          -- none of the following
                regular1822(2),
                hdh1822(3),
                ddn-x25(4),
                rfc877-x25(5),
                ethernet-csmacd(6),
                iso88023-csmacd(7),
                iso88024-tokenBus(8),
                iso88025-tokenRing(9),
                iso88026-man(10),
                starLan(11),
                proteon-10Mbit(12),
                proteon-80Mbit(13),
                hyperchannel(14),
                fddi(15),
                lapb(16),
                sdlc(17),
                ds1(18),           -- T-1
                e1(19),            -- european equiv. of T-1
                basicISDN(20),
                primaryISDN(21),   -- proprietary serial
                propPointToPointSerial(22),
                ppp(23),
```

```
                softwareLoopback(24),
                eon(25),            -- CLNP over IP [11]
                ethernet-3Mbit(26),
                nsip(27),           -- XNS over IP
                slip(28),           -- generic SLIP
                ultra(29),          -- ULTRA technologies
                ds3(30),            -- T-3
                sip(31),            -- SMDS
                frame-relay(32)
            }
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The type of interface, distinguished according to
            the physical/link protocol(s) immediately `below'
            the network layer in the protocol stack."
    ::= { ifEntry 3 }

ifMtu OBJECT-TYPE
    SYNTAX  INTEGER
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The size of the largest datagram which can be
            sent/received on the interface, specified in
            octets.  For interfaces that are used for
            transmitting network datagrams, this is the size
            of the largest network datagram that can be sent
            on the interface."
    ::= { ifEntry 4 }

ifSpeed OBJECT-TYPE
    SYNTAX  Gauge
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "An estimate of the interface's current bandwidth
            in bits per second.  For interfaces which do not
            vary in bandwidth or for those where no accurate
            estimation can be made, this object should contain
            the nominal bandwidth."
    ::= { ifEntry 5 }

ifPhysAddress OBJECT-TYPE
    SYNTAX  PhysAddress
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The interface's address at the protocol layer
            immediately `below' the network layer in the
            protocol stack.  For interfaces which do not have
            such an address (e.g., a serial line), this object
            should contain an octet string of zero length."
    ::= { ifEntry 6 }

ifAdminStatus OBJECT-TYPE
    SYNTAX  INTEGER {
                up(1),       -- ready to pass packets
                down(2),
                testing(3)   -- in some test mode
            }
    ACCESS  read-write
    STATUS  mandatory
    DESCRIPTION
```

```
            "The desired state of the interface.  The
            testing(3) state indicates that no operational
            packets can be passed."
    ::= { ifEntry 7 }

ifOperStatus OBJECT-TYPE
    SYNTAX  INTEGER {
                up(1),       -- ready to pass packets
                down(2),
                testing(3)   -- in some test mode
            }
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The current operational state of the interface.
            The testing(3) state indicates that no operational
            packets can be passed."
    ::= { ifEntry 8 }

ifLastChange OBJECT-TYPE
    SYNTAX  TimeTicks
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The value of sysUpTime at the time the interface
            entered its current operational state.  If the
            current state was entered prior to the last re-
            initialization of the local network management
            subsystem, then this object contains a zero
            value."
    ::= { ifEntry 9 }

ifInOctets OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of octets received on the
            interface, including framing characters."
    ::= { ifEntry 10 }

ifInUcastPkts OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of subnetwork-unicast packets
            delivered to a higher-layer protocol."
    ::= { ifEntry 11 }

ifInNUcastPkts OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of non-unicast (i.e., subnetwork-
            broadcast or subnetwork-multicast) packets
            delivered to a higher-layer protocol."
    ::= { ifEntry 12 }

ifInDiscards OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
```

```
    DESCRIPTION
            "The number of inbound packets which were chosen
            to be discarded even though no errors had been
            detected to prevent their being deliverable to a
            higher-layer protocol.  One possible reason for
            discarding such a packet could be to free up
            buffer space."
    ::= { ifEntry 13 }

ifInErrors OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of inbound packets that contained
            errors preventing them from being deliverable to a
            higher-layer protocol."
    ::= { ifEntry 14 }

ifInUnknownProtos OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The number of packets received via the interface
            which were discarded because of an unknown or
            unsupported protocol."
    ::= { ifEntry 15 }

ifOutOctets OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of octets transmitted out of the
            interface, including framing characters."
    ::= { ifEntry 16 }

ifOutUcastPkts OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of packets that higher-level
            protocols requested be transmitted to a
            subnetwork-unicast address, including those that
            were discarded or not sent."
    ::= { ifEntry 17 }

ifOutNUcastPkts OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
    STATUS  mandatory
    DESCRIPTION
            "The total number of packets that higher-level
            protocols requested be transmitted to a non-
            unicast (i.e., a subnetwork-broadcast or
            subnetwork-multicast) address, including those
            that were discarded or not sent."
    ::= { ifEntry 18 }

ifOutDiscards OBJECT-TYPE
    SYNTAX  Counter
    ACCESS  read-only
```

```
        STATUS  mandatory
        DESCRIPTION
                "The number of outbound packets which were chosen
                to be discarded even though no errors had been
                detected to prevent their being transmitted.  One
                possible reason for discarding such a packet could
                be to free up buffer space."
        ::= { ifEntry 19 }

    ifOutErrors OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The number of outbound packets that could not be
                transmitted because of errors."
        ::= { ifEntry 20 }

    ifOutQLen OBJECT-TYPE
        SYNTAX  Gauge
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "The length of the output packet queue (in
                packets)."
        ::= { ifEntry 21 }

    ifSpecific OBJECT-TYPE
        SYNTAX  OBJECT IDENTIFIER
        ACCESS  read-only
        STATUS  mandatory
        DESCRIPTION
                "A reference to MIB definitions specific to the
                particular media being used to realize the
                interface.  For example, if the interface is
                realized by an ethernet, then the value of this
                object refers to a document defining objects
                specific to ethernet.  If this information is not
                present, its value should be set to the OBJECT
                IDENTIFIER { 0 0 }, which is a syntatically valid
                object identifier, and any conformant
                implementation of ASN.1 and BER must be able to
                generate and recognize this value."
        ::= { ifEntry 22 }
```

Приложение 2: Идентификаторы объектов (OID) в MIB и их обозначение.

Название объекта	Цифро-точечное представление
Org	1.3
Dod	1.3.6
Internet	1.3.6.1
directory	1.3.6.1.1
mgmt	1.3.6.1.2
experimental	1.3.6.1.3
private	1.3.6.1.4
enterprises	1.3.6.1.4.1
security	1.3.6.1.5
snmpV2	1.3.6.1.6
snmpDomains	1.3.6.1.6.1
snmpProxys	1.3.6.1.6.2
snmpModules	1.3.6.1.6.3
snmpMIB	1.3.6.1.6.3.1
snmpMIBObjects	1.3.6.1.6.3.1.1
snmpTraps	1.3.6.1.6.3.1.1.5
mib-2	1.3.6.1.2.1
ifMIB	1.3.6.1.2.1.31
interfaces	1.3.6.1.2.1.2
ifMIBObjects	1.3.6.1.2.1.31.1
ifConformance	1.3.6.1.2.1.31.2
ifTableLastChange	1.3.6.1.2.1.31.1.5
ifXTable	1.3.6.1.2.1.31.1.1
ifStackTable	1.3.6.1.2.1.31.1.2
ifStackLastChange	1.3.6.1.2.1.31.1.6
ifRcvAddressTable	1.3.6.1.2.1.31.1.4
ifTestTable	1.3.6.1.2.1.31.1.3
ifXEntry	1.3.6.1.2.1.31.1.1.1
ifName	1.3.6.1.2.1.31.1.1.1.1
ifInMulticastPkts	1.3.6.1.2.1.31.1.1.1.2
ifInBroadcastPkts	1.3.6.1.2.1.31.1.1.1.3
ifOutMulticastPkts	1.3.6.1.2.1.31.1.1.1.4
ifOutBroadcastPkts	1.3.6.1.2.1.31.1.1.1.5
ifLinkUpDownTrapEnable	1.3.6.1.2.1.31.1.1.1.14
ifHighSpeed	1.3.6.1.2.1.31.1.1.1.15
ifPromiscuousMode	1.3.6.1.2.1.31.1.1.1.16
ifConnectorPresent	1.3.6.1.2.1.31.1.1.1.17
ifAlias	1.3.6.1.2.1.31.1.1.1.18
ifCounterDiscontinuityTime	1.3.6.1.2.1.31.1.1.1.19
ifStackEntry	1.3.6.1.2.1.31.1.2.1
ifStackHigherLayer	1.3.6.1.2.1.31.1.2.1.1
ifStackLowerLayer	1.3.6.1.2.1.31.1.2.1.2
ifStackStatus	1.3.6.1.2.1.31.1.2.1.3
ifRcvAddressEntry	1.3.6.1.2.1.31.1.4.1
ifRcvAddressAddress	1.3.6.1.2.1.31.1.4.1.1
ifRcvAddressStatus	1.3.6.1.2.1.31.1.4.1.2
ifRcvAddressType	1.3.6.1.2.1.31.1.4.1.3
ifTestEntry	1.3.6.1.2.1.31.1.3.1
ifTestId	1.3.6.1.2.1.31.1.3.1.1
ifTestStatus	1.3.6.1.2.1.31.1.3.1.2
ifTestType	1.3.6.1.2.1.31.1.3.1.3
ifTestResult	1.3.6.1.2.1.31.1.3.1.4
ifTestCode	1.3.6.1.2.1.31.1.3.1.5
ifTestOwner	1.3.6.1.2.1.31.1.3.1.6
ifGroups	1.3.6.1.2.1.31.2.1
ifCompliances	1.3.6.1.2.1.31.2.2
ifGeneralInformationGroup	1.3.6.1.2.1.31.2.1.10

ifFixedLengthGroup	1.3.6.1.2.1.31.2.1.2
ifHCFixedLengthGroup	1.3.6.1.2.1.31.2.1.3
ifPacketGroup	1.3.6.1.2.1.31.2.1.4
ifHCPacketGroup	1.3.6.1.2.1.31.2.1.5
ifVHCPacketGroup	1.3.6.1.2.1.31.2.1.6
ifRcvAddressGroup	1.3.6.1.2.1.31.2.1.7
ifStackGroup2	1.3.6.1.2.1.31.2.1.11
ifCounterDiscontinuityGroup	1.3.6.1.2.1.31.2.1.13
ifGeneralGroup	1.3.6.1.2.1.31.2.1.1
ifTestGroup	1.3.6.1.2.1.31.2.1.8
ifStackGroup	1.3.6.1.2.1.31.2.1.9
ifOldObjectsGroup	1.3.6.1.2.1.31.2.1.12
ifCompliance2	1.3.6.1.2.1.31.2.2.2
ifCompliance	1.3.6.1.2.1.31.2.2.1
ifNumber	1.3.6.1.2.1.2.1
ifTable	1.3.6.1.2.1.2.2
ifEntry	1.3.6.1.2.1.2.2.1
ifIndex	1.3.6.1.2.1.2.2.1.1
ifDescr	1.3.6.1.2.1.2.2.1.2
ifType	1.3.6.1.2.1.2.2.1.3
ifMtu	1.3.6.1.2.1.2.2.1.4
ifSpeed	1.3.6.1.2.1.2.2.1.5
ifPhysAddress	1.3.6.1.2.1.2.2.1.6
ifAdminStatus	1.3.6.1.2.1.2.2.1.7
ifOperStatus	1.3.6.1.2.1.2.2.1.8
ifLastChange	1.3.6.1.2.1.2.2.1.9
ifInOctets	1.3.6.1.2.1.2.2.1.10
ifInUcastPkts	1.3.6.1.2.1.2.2.1.11
ifInNUcastPkts	1.3.6.1.2.1.2.2.1.12
ifInDiscards	1.3.6.1.2.1.2.2.1.13
ifInErrors	1.3.6.1.2.1.2.2.1.14
ifInUnknownProtos	1.3.6.1.2.1.2.2.1.15
ifOutOctets	1.3.6.1.2.1.2.2.1.16
ifOutUcastPkts	1.3.6.1.2.1.2.2.1.17
ifOutNUcastPkts	1.3.6.1.2.1.2.2.1.18
ifOutDiscards	1.3.6.1.2.1.2.2.1.19
ifOutErrors	1.3.6.1.2.1.2.2.1.20
ifOutQLen	1.3.6.1.2.1.2.2.1.21
ifSpecific	1.3.6.1.2.1.2.2.1.22

Приложение 3: Коды вендоров сетевого оборудования

ETHERNET VENDOR ADDRESS COMPONENTS

Ethernet hardware addresses are 48 bits, expressed as 12 hexadecimal digits (0-9, plus A-F, capitalized). These 12 hex digits consist of the **first/left 6 digits** (which should match **the vendor of the Ethernet interface** within the station) and the **last/right 6 digits** which specify the interface **serial number for that interface vendor**.

Ethernet addresses might be written unhyphenated (e.g., 123456789ABC), or with one hyphen (e.g., 123456-789ABC), but should be written hyphenated by octets (e.g., **12-34-56-78-9A-BC**).

These addresses are physical station addresses, not multicast nor broadcast, so the second hex digit (reading from the left) will be even, not odd.

At present, it is not clear how the IEEE assigns Ethernet block addresses. Whether in blocks of 2**24 or 2**25, and whether multicasts are assigned with that block or separately. A portion of the vendor block address is reportedly assigned serially, with the other portion intentionally assigned randomly. If there is a global algorithm for which addresses are designated to be physical (in a chipset) versus logical (assigned in software), or globally-assigned versus locally-assigned addresses, some of the known addresses do not follow the scheme (e.g., AA0003; 02xxxx).

00000C	**Cisco**	
00000E	**Fujitsu**	
00000F	NeXT	
000010	Sytek	
00001D	**Cabletron**	
000020	DIAB (Data Intdustrier AB)	
000022	Visual Technology	
00002A	TRW	
000032	GPT Limited (reassigned from GEC Computers Ltd)	
00005A	S & Koch	
00005E	**IANA**	
000065	Network General	
00006B	MIPS	
000077	MIPS	
00007A	Ardent	
000089	Cayman Systems	Gatorbox
000093	Proteon	
00009F	Ameristar Technology	
0000A2	Wellfleet	
0000A3	Network Application Technology	
0000A6	Network General (internal assignment, not for products)	
0000A7	NCD	X-terminals
0000A9	Network Systems	
0000AA	**Xerox**	**Xerox machines**
0000B3	CIMLinc	
0000B7	Dove	Fastnet
0000BC	Allen-Bradley	

0000C0 Western Digital
0000C5 Farallon phone net card
0000C6 HP Intelligent Networks Operation (formerly Eon Systems)
0000C8 Altos
0000C9 Emulex   Terminal Servers
0000D7 Dartmouth College (NED Router)
0000D8 3Com? Novell?   PS/2
0000DD Gould
0000DE Unigraph
0000E2 Acer Counterpoint
0000EF Alantec
0000FD High Level Hardvare (Orion, UK)
000102 BBN   BBN internal usage (not registered)
0020AF 3COM ???
001700 Kabel
008064 Wyse Technology / Link Technologies
00802B IMAC ???
00802D Xylogics, Inc. Annex terminal servers
00808C Frontier Software Development
0080C2 IEEE 802.1 Committee
0080D3 Shiva
00AA00 Intel
00DD00 Ungermann-Bass
00DD01 Ungermann-Bass
020701 Racal InterLan
020406 BBN   BBN internal usage (not registered)
026086 Satelcom MegaPac (UK)
02608C 3Com   IBM PC; Imagen; Valid; Cisco
02CF1F CMC   Masscomp; Silicon Graphics; Prime EXL
080002 3Com (Formerly Bridge)
080003 ACC (Advanced Computer Communications)
080005 Symbolics   Symbolics LISP machines
080008 BBN
080009 Hewlett-Packard
08000A Nestar Systems
08000B Unisys
080011 Tektronix, Inc.
080014 Excelan   BBN Butterfly, Masscomp, Silicon Graphics
080017 NSC
08001A Data General
08001B Data General
08001E Apollo
080020 Sun   Sun machines
080022 NBI
080025 CDC
080026 Norsk Data (Nord)
080027 PCS Computer Systems GmbH
080028 TI   Explorer
08002B DEC
08002E Metaphor
08002F Prime Computer Prime 50-Series LHC300
080036 Intergraph   CAE stations
080037 Fujitsu-Xerox
080038 Bull
080039 Spider Systems
080041 DCA Digital Comm. Assoc.
080045 ???? (maybe Xylogics, but they claim not to know this number)
080046 Sony

080047	Sequent	
080049	Univation	
08004C	Encore	
08004E	BICC	
080056	Stanford University	
080058	???	DECsystem-20
08005A	**IBM**	
080067	Comdesign	
080068	Ridge	
080069	Silicon Graphics	
08006E	Concurrent	Masscomp
080075	DDE (Danish Data Elektronik A/S)	
08007C	Vitalink	TransLAN III
080080	XIOS	
080086	Imagen/QMS	
080087	Xyplex	terminal servers
080089	Kinetics	AppleTalk-Ethernet interface
08008B	Pyramid	
08008D	XyVision	XyVision machines
080090	Retix Inc	Bridges
484453	HDS ???	
800010	**AT&T**	
AA0000	**DEC**	**obsolete**
AA0001	DEC	obsolete
AA0002	DEC	obsolete
AA0003	DEC	Global physical address for some DEC machines
AA0004	DEC	Local logical address for systems running DECNET

Приложение 4: Коды типов протоколов, обслуживаемых протоколом Ethernet

ETHER TYPES

Many of the networks of all classes are Ethernets (10Mb) or Experimental Ethernets (3Mb). These systems use a message "type" field in much the same way the ARPANET uses the "link" field.

If you need an Ether Type, contact:
Xerox Systems Institute
3400 Hillview Ave.
PO BOX 10034
Palo Alto, CA 94303
Phone: 415-813-7164
Contact: Fonda Lix Pallone

The following list of **EtherTypes** is contributed unverified information from various sources.

Assignments:

Ethernet		Exp. Ethernet		Description	References
decimal	Hex	decimal	octal		
000	0000-05DC	-	-	IEEE802.3 Length Field	[XEROX]
257	0101-01FF	-	-	Experimental	[XEROX]
512	0200	512	1000	XEROX PUP (see 0A00)	[8,XEROX]
513	0201	-	-	PUP Addr Trans (see 0A01)	[XEROX]
	0400			Nixdorf	[XEROX]
1536	0600	1536	3000	XEROX NS IDP	[133,XEROX]
	0660			DLOG	[XEROX]
	0661			DLOG	[XEROX]
2048	**0800**	**513**	**1001**	**Internet IP (IPv4)**	**[105,JBP]**
2049	0801	-	-	X.75 Internet	[XEROX]
2050	0802	-	-	NBS Internet	[XEROX]
2051	0803	-	-	ECMA Internet	[XEROX]
2052	0804	-	-	Chaosnet	[XEROX]
2053	**0805**	**-**	**-**	**X.25 Level 3**	**[XEROX]**
2054	**0806**	**-**	**-**	**ARP**	**[88,JBP]**
2055	0807	-	-	XNS Compatability	[XEROX]
2076	081C	-	-	Symbolics Private	[DCP1]
2184	0888-088A	-	-	Xyplex	[XEROX]
2304	0900	-	-	Ungermann-Bass net debugr	[XEROX]
2560	0A00	-	-	Xerox IEEE802.3 PUP	[XEROX]
2561	0A01	-	-	PUP Addr Trans	[XEROX]
2989	0BAD	-	-	Banyan Systems	[XEROX]
4096	1000	-	-	Berkeley Trailer nego	[XEROX]
4097	1001-100F	-	-	Berkeley Trailer encap/IP	[XEROX]
5632	1600	-	-	Valid Systems	[XEROX]
16962	4242	-	-	PCS Basic Block Protocol	[XEROX]
21000	5208	-	-	BBN Simnet	[XEROX]
24576	6000	-	-	DEC Unassigned (Exp.)	[XEROX]
24577	6001	-	-	DEC MOP Dump/Load	[XEROX]
24578	6002	-	-	DEC MOP Remote Console	[XEROX]
24579	6003	-	-	DEC DECNET Phase IV Route	[XEROX]
24580	6004	-	-	DEC LAT	[XEROX]
24581	6005	-	-	DEC Diagnostic Protocol	[XEROX]
24582	6006	-	-	DEC Customer Protocol	[XEROX]

24583	6007	-	-	DEC LAVC, SCA	[XEROX]
24584	6008-6009	-	-	DEC Unassigned	[XEROX]
24586	**6010-6014**	**-**	**-**	**3Com Corporation**	**[XEROX]**
28672	7000	-	-	Ungermann-Bass download	[XEROX]
28674	7002	-	-	Ungermann-Bass dia/loop	[XEROX]
28704	7020-7029	-	-	LRT	[XEROX]
28720	7030	-	-	Proteon	[XEROX]
28724	7034	-	-	Cabletron	[XEROX]
32771	8003	-	-	Cronus VLN	[131,DT15]
32772	8004	-	-	Cronus Direct	[131,DT15]
32773	8005	-	-	HP Probe	[XEROX]
32774	8006	-	-	Nestar	[XEROX]
32776	8008	-	-	AT&T	[XEROX]
32784	8010	-	-	Excelan	[XEROX]
32787	8013	-	-	SGI diagnostics	[AXC]
32788	8014	-	-	SGI network games	[AXC]
32789	8015	-	-	SGI reserved	[AXC]
32790	8016	-	-	SGI bounce server	[AXC]
32793	8019	-	-	Apollo Computers	[XEROX]
32815	802E	-	-	Tymshare	[XEROX]
32816	802F	-	-	Tigan, Inc.	[XEROX]
32821	8035	-	-	Reverse ARP	[48,JXM]
32822	8036	-	-	Aeonic Systems	[XEROX]
32824	8038	-	-	DEC LANBridge	[XEROX]
32825	**8039-803C**	**-**	**-**	**DEC Unassigned**	**[XEROX]**
32829	803D	-	-	DEC Ethernet Encryption	[XEROX]
32830	803E	-	-	DEC Unassigned	[XEROX]
32831	803F	-	-	DEC LAN Traffic Monitor	[XEROX]
32832	8040-8042	-	-	DEC Unassigned	[XEROX]
32836	8044	-	-	Planning Research Corp.	[XEROX]
32838	8046	-	-	AT&T	[XEROX]
32839	8047	-	-	AT&T	[XEROX]
32841	8049	-	-	ExperData	[XEROX]
32859	805B	-	-	Stanford V Kernel exp.	[XEROX]
32860	805C	-	-	Stanford V Kernel prod.	[XEROX]
32861	805D	-	-	Evans & Sutherland	[XEROX]
32864	8060	-	-	Little Machines	[XEROX]
32866	8062	-	-	Counterpoint Computers	[XEROX]
32869	8065	-	-	Univ. of Mass. @ Amherst	[XEROX]
32870	8066	-	-	Univ. of Mass. @ Amherst	[XEROX]
32871	8067	-	-	Veeco Integrated Auto.	[XEROX]
32872	8068	-	-	General Dynamics	[XEROX]
32873	8069	-	-	AT&T	[XEROX]
32874	806A	-	-	Autophon	[XEROX]
32876	806C	-	-	ComDesign	[XEROX]
32877	806D	-	-	Computgraphic Corp.	[XEROX]
32878	806E-8077	-	-	Landmark Graphics Corp.	[XEROX]
32890	807A	-	-	Matra	[XEROX]
32891	807B	-	-	Dansk Data Elektronik	[XEROX]
32892	807C	-	-	Merit Internodal	[HWB]
32893	807D-807F	-	-	Vitalink Communications	[XEROX]
32896	8080	-	-	Vitalink TransLAN III	[XEROX]
32897	8081-8083	-	-	Counterpoint Computers	[XEROX]
32923	809B	-	-	Appletalk	[XEROX]
32924	809C-809E	-	-	Datability	[XEROX]
32927	809F	-	-	Spider Systems Ltd.	[XEROX]
32931	80A3	-	-	Nixdorf Computers	[XEROX]
32932	80A4-80B3	-	-	Siemens Gammasonics Inc.	[XEROX]

32960	80C0-80C3	-	-	DCA Data Exchange Cluster	[XEROX]
	80C4			Banyan Systems	[XEROX]
	80C5			Banyan Systems	[XEROX]
32966	80C6	-	-	Pacer Software	[XEROX]
32967	80C7	-	-	Applitek Corporation	[XEROX]
32968	80C8-80CC	-	-	Intergraph Corporation	[XEROX]
32973	80CD-80CE	-	-	Harris Corporation	[XEROX]
32975	80CF-80D2	-	-	Taylor Instrument	[XEROX]
32979	80D3-80D4	-	-	Rosemount Corporation	[XEROX]
32981	80D5	-	-	IBM SNA Service on Ether	[XEROX]
32989	80DD	-	-	Varian Associates	[XEROX]
32990	80DE-80DF	-	-	Integrated Solutions TRFS	[XEROX]
32992	80E0-80E3	-	-	Allen-Bradley	[XEROX]
32996	80E4-80F0	-	-	Datability	[XEROX]
33010	80F2	-	-	Retix	[XEROX]
33011	80F3	-	-	AppleTalk AARP (Kinetics)	[XEROX]
33012	80F4-80F5	-	-	Kinetics	[XEROX]
33015	80F7	-	-	Apollo Computer	[XEROX]
33023	80FF-8103	-	-	Wellfleet Communications	[XEROX]
33031	8107-8109	-	-	Symbolics Private	[XEROX]
33072	8130	-	-	Hayes Microcomputers	[XEROX]
33073	8131	-	-	VG Laboratory Systems	[XEROX]
	8132-8136			Bridge Communications	[XEROX]
33079	8137-8138	-	-	Novell, Inc.	[XEROX]
33081	8139-813D	-	-	KTI	[XEROX]
	8148			Logicraft	[XEROX]
	8149			Network Computing Devices	[XEROX]
	814A			Alpha Micro	[XEROX]
33100	**814C**	**-**	**-**	**SNMP**	**[JKR1]**
	814D			BIIN	[XEROX]
	814E			BIIN	[XEROX]
	814F			Technically Elite Concept	[XEROX]
	8150			Rational Corp	[XEROX]
	8151-8153			**Qualcomm**	**[XEROX]**
	815C-815E			Computer Protocol Pty Ltd	[XEROX]
	8164-8166			Charles River Data System	[XEROX]
	817D-818C			Protocol Engines	[XEROX]
	818D			**Motorola Computer**	**[XEROX]**
	819A-81A3			Qualcomm	[XEROX]
	81A4			ARAI Bunkichi	[XEROX]
	81A5-81AE			RAD Network Devices	[XEROX]
	81B7-81B9			Xyplex	[XEROX]
	81CC-81D5			Apricot Computers	[XEROX]
	81D6-81DD			Artisoft	[XEROX]
	81E6-81EF			Polygon	[XEROX]
	81F0-81F2			Comsat Labs	[XEROX]
	81F3-81F5			SAIC	[XEROX]
	81F6-81F8			VG Analytical	[XEROX]
	8203-8205			Quantum Software	[XEROX]
	8221-8222			Ascom Banking Systems	[XEROX]
	823E-8240			Advanced Encryption Syste	[XEROX]
	827F-8282			Athena Programming	[XEROX]
	8263-826A			Charles River Data System	[XEROX]
	829A-829B			Inst Ind Info Tech	[XEROX]
	829C-82AB			Taurus Controls	[XEROX]
	82AC-8693			Walker Richer & Quinn	[XEROX]
	8694-869D			Idea Courier	[XEROX]
	869E-86A1			Computer Network Tech	[XEROX]

	86A3-86AC			Gateway Communications	[XEROX]
	86DB			SECTRA	[XEROX]
	86DE			Delta Controls	[XEROX]
34543	86DF	-	-	ATOMIC	[JBP]
	86E0-86EF			Landis & Gyr Powers	[XEROX]
	8700-8710			Motorola	[XEROX]
	8A96-8A97			Invisible Software	[XEROX]
36864	9000	-	-	Loopback	[XEROX]
36865	9001	-	-	3Com(Bridge) XNS Sys Mgmt	[XEROX]
36866	9002	-	-	3Com(Bridge) TCP-IP Sys	[XEROX]
36867	9003	-	-	3Com(Bridge) loop detect	[XEROX]
65280	FF00	-	-	BBN VITAL-LanBridge cache	[XEROX]
	FF00-FF0F			ISC Bunker Ramo	[XEROX]

The standard for transmission of IP datagrams over Ethernets and Experimental Ethernets is specified in [RFC894] and [RFC895] respectively.

NOTE: Ethernet 48-bit address blocks are assigned by the IEEE.

IEEE Registration Authority
c/o Iris Ringel
IEEE Standards Department
445 Hoes Lane, P.O. Box 1331
Piscataway, NJ 08855-1331
Phone +1 908 562 3813
Fax: +1 908 562 1571

Анатолий Егорович Костюкович

Практикум к контрольной работе: «Организация сети для услуг управления».

Редактор: Ромашова Т.И..
Корректор:

подписано в печать.....
сдано в набор ....., бумага писчая N1.

Формат бумаги 60x84/16. Шрифт N10, печать РИЗО,
изд. листов ..... Тираж 500, заказ N .....

Сибирский государственный университет телекоммуникаций и информатики
(ФГОБУ ВПО «СибГУТИ»)

630102, Новосибирск, ул.Кирова,86